my spring 2011, EME 121 web page
UC Davis, Engineering Mechanical dept

Nasser M. Abbasi

June 9, 2011 Compiled on December 9, 2018 at 5:43pm [public]

Contents

Chapter 1

Introduction

I took this course in spring 2011 to learn more about dynamics. The instructor is Professor Margolis, UC Davis. A very helpful Professor who has lots of knowledge in the subject. His exams are reasonable but needed to study hard for them. He makes the subject very interesting and explains hard thing in simple way to make it easy to understand. I recommend his courses.

1.1 Syllabus

EME-121 Engineering Applications of Dynamics

Professor Margolis, Room 2010 Bainer Hall, Spring 2011

Lecture TTh 9:00-10:20 Room 1062 Bainer Hall
Lab meeting to be announced

Week	Date-Mon	Assignment	Problems
1	3/28	Newton's Laws, 1.1-1.4	1.1, 1.2, 1.4, 1.5, 1.6
2	4/4	Newton's Laws, 1.5-1.7	1.11, 1.12, 1.13, 1.15, 1.18
3	4/11	Equations of Motion, 2.1-2.6	2.1, 2.2, 2.5, 2.6, 2.7, 2.8, 2.13, 2.19
4	4/18	Energy and Lagrange, 4.1-4.4	4.1, 4.3, 4.4, 4.6, 4.8
5	4/25	Energy and Lagrange, 4.5-4.8	4.9, 4.10, 4.11, 4.12, 4.13, 4.17
6	5/2	Vehicle Dynamics, 5.1-5.7	5.1, 5.2, 5.5, 5.7, 5.8
7	5/9	Rigid Bodies in 3d, 7.1-7.2	7.2, 7.6, 7.7, 7.8, 7.11
8	5/16	Rigid Bodies in 3d, 7.3-7.4	
9	5/23	Vibrations, 8.1-8.4	8.1, 8.3, 8.6, 8.10, 8.18
10	5/30	Vibrations and Special Topics	

Required text:

Engineering Applications of Dynamics, Wiley and sons, 2008
Karnopp and Margolis

Grading: 1 Final test 40%
1 Midterm 40%
Simulations, homework 20%

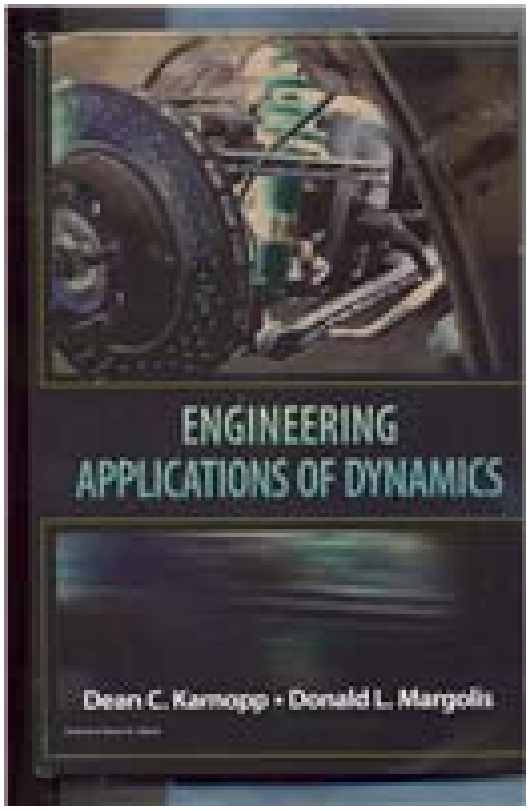
Final Examination, Wednesday June 8, 1:00-3:00 PM

1.1.1 course description from catalog

Prerequisite: Engineering 102. Technical elective that revisits dynamic principles with emphasis on engineering applications; stressing importance of deriving equations of motion and setting these into format for computer solution with computer simulation lab, students gain experience with solving complex, real engineering applications. III. (III.) Karnopp, Margolis

1.1.2 Text book

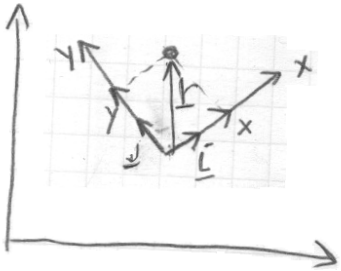
Engineering Applications of Dynamics, by Dean C. Karnopp (Author), Donald L. Margolis (Author)



A very good book on Engineering dynamics, with lots of worked examples and nice diagrams.

1.2 Teacher note on vector rate of change

RATE OF CHANGE OF A VECTOR REFERRED TO A MOVING FRAME



x, y a moving frame

$\underline{i}, \underline{j}$ unit vectors in the moving frame, but they change directions if frame rotates

$$\underline{r} = x\underline{i} + y\underline{j}$$

$$\dot{\underline{r}} = \frac{d}{dt}\underline{r} = \dot{x}\underline{i} + \dot{y}\underline{j} + x\dot{\underline{i}} + y\dot{\underline{j}}$$

Just like $\underline{\epsilon}_r$ and $\underline{\epsilon}_\theta$,

$$\dot{\underline{i}} = \underline{\omega} \times \underline{i}$$

$$\dot{\underline{j}} = \underline{\omega} \times \underline{j}$$

$$\therefore \dot{\underline{r}} = \dot{x}\underline{i} + \dot{y}\underline{j} + x(\underline{\omega} \times \underline{i}) + y(\underline{\omega} \times \underline{j})$$

$$\dot{\underline{r}} = \underbrace{\dot{x}\underline{i} + \dot{y}\underline{j}}_{\text{change in length}} + \underline{\omega} \times \underbrace{(x\underline{i} + y\underline{j})}_{\underline{r}}$$

since $\underline{r}$ could be any vector in the x, y frame

$$\frac{d\underline{A}}{dt} = \left.\frac{\partial \underline{A}}{\partial t}\right|_{rel} + \underline{\omega} \times \underline{A}$$

Chapter 2

exams

2.1 my typed solution of EME 121 midterm

This is my EME 121 midterm post-mortem write up.

2.1.1 Problem 1

EME 121 Engineering Applications of Dynamics
Midterm Exam, Spring 2011
Professor Margolis
1 1/2 hours, open book and notes

Name_____

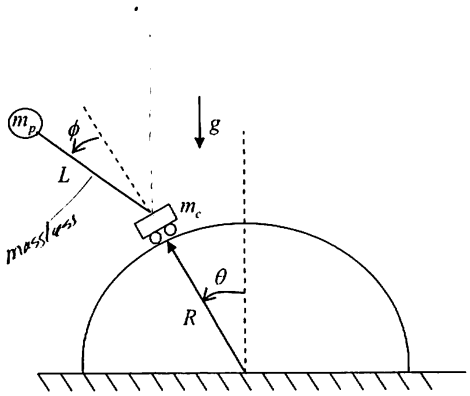
Problem 1 34 points _____

Problem 2 33 points _____

Problem 3 33 points _____

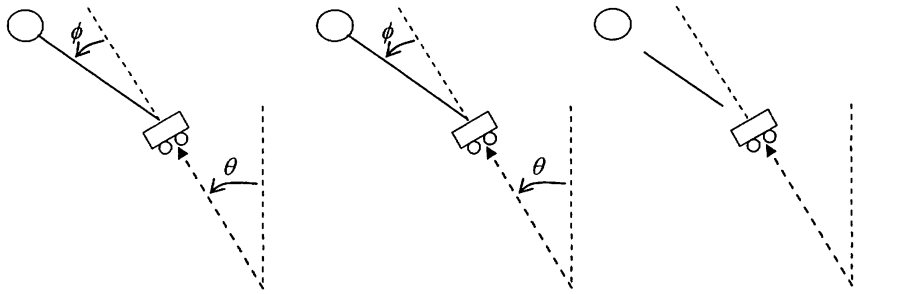
Problem 1

A cart of mass m_c rolls without slip on a semi-circular guide of radius R. The cart always remains on the surface of the guide. Pinned without friction to the cart is a pendulum of length L and mass m. Gravity acts vertically downward. The coordinate θ is measured from an inertial vertical line and θ tracks the position of the cart. The variable ϕ is measured relative to θ. We desire the equations of motion for this system in terms of the coordinates shown on the figure.



To assist you in your derivation the schematics below are provided.

A. Using arrows and symbols show velocity and acceleration components. If you use the transfer velocity and acceleration formulas, clearly show the axes you are using and indicate which component in your diagram corresponds to which component in the formulas. Also use arrows and symbols to indicate the forces on the force diagram.



Velocity diagram acceleration diagram force diagram

B. Using Newton's laws derive the equations of motion and insure that there are sufficient equations for the number of unknown time-varying quantities.

C. Put your result into first order form suitable for simulation.

Cart moves on top of circle as shown, with pendulum attached. Find equations of motion. This is 2 DOF, using θ and ϕ for generalized coordinates.

2.1.2 Velocity, acceleration and force diagrams

For the mass m we use $V_p = V_o + \omega \times r + V_{rel}$. The important thing to see is that ω of the frame of reference for m is relative to the inertial frame always. Hence $\omega = \dot{\phi} + \dot{\theta}$. In this problem, we see that $V_{rel} = 0$ for mass m since the length L do not change. Therefore, for m we have

$$\begin{aligned} V_p &= V_o + \omega \times r + V_{rel} \\ &= R\dot{\theta} + L\left(\dot{\phi} + \dot{\theta}\right) \end{aligned}$$

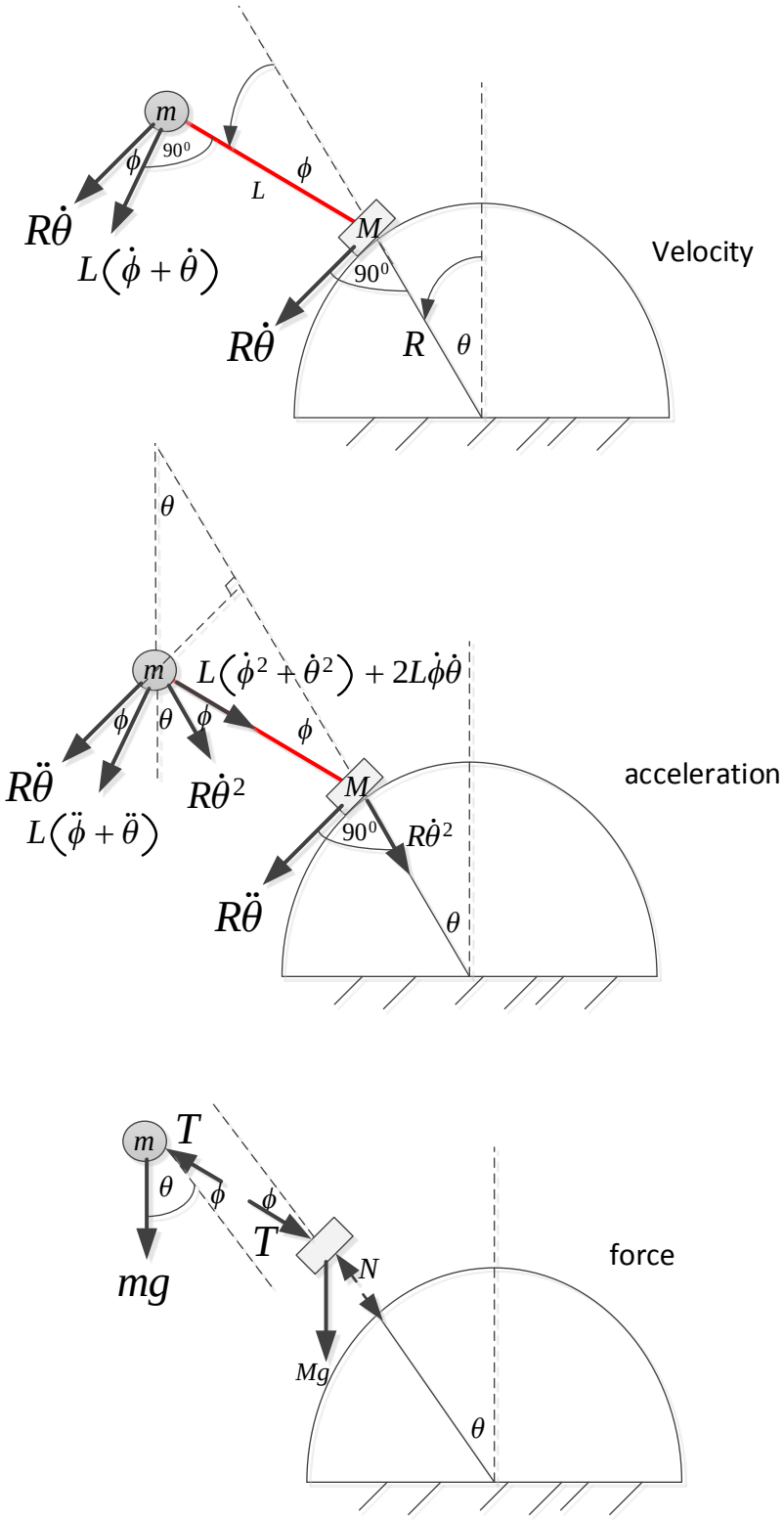
For the acceleration, we have

$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times V_{rel} + a_{rel}$$

For the mass m, V_{rel} and a_{rel} are zero, since the mass m is on a fixed rod. So we have

$$\begin{aligned} a_p &= a_0 + \left(\ddot{\phi} + \ddot{\theta}\right) \times r + \left(\dot{\phi} + \dot{\theta}\right) \times \left(\left(\dot{\phi} + \dot{\theta}\right) \times r\right) \\ &= R\ddot{\theta} + L\left(\ddot{\phi} + \ddot{\theta}\right) + \left(\dot{\phi} + \dot{\theta}\right)\left(\dot{\phi} + \dot{\theta}\right) L \\ &= R\ddot{\theta} + L\left(\ddot{\phi} + \ddot{\theta}\right) + L\dot{\phi}^2 + L\dot{\theta}^2 + 2L\dot{\phi}\dot{\theta} \\ &= R\ddot{\theta} + L\left(\ddot{\phi} + \ddot{\theta}\right) + L\left(\dot{\phi}^2 + \dot{\theta}^2\right) + 2L\dot{\phi}\dot{\theta} \end{aligned}$$

So, in diagram, this is how it looks



The unknowns are θ, ϕ, T, N, hence we need 4 equations.

2.1.3 Solution using $F = ma$

For pendulum mass m, resolving forces radially we obtain

$$\begin{aligned} mg\cos(\theta+\phi) - T &= m\left(L\left(\dot{\phi}^2+\dot{\theta}^2\right) + 2L\dot{\phi}\dot{\theta} + R\dot{\theta}^2\cos\phi - R\ddot{\theta}\sin\phi\right) \\ T &= m\left[g\cos(\theta+\phi) - L\left(\dot{\phi}^2+\dot{\theta}^2\right) - 2L\dot{\phi}\dot{\theta} - R\dot{\theta}^2\cos\phi + R\ddot{\theta}\sin\phi\right] \end{aligned} \tag{1}$$

and resolving forces tangentially gives

$$\begin{aligned} mg\sin(\theta+\phi) &= m\left(L\left(\ddot{\phi}+\ddot{\theta}\right) + R\dot{\theta}^2\sin\phi + R\ddot{\theta}\cos\phi\right) \\ \ddot{\phi} &= \frac{g}{L}\sin(\theta+\phi) - \frac{R}{L}\left(\dot{\theta}^2\sin\phi + \ddot{\theta}\cos\phi\right) - L\ddot{\theta} \end{aligned} \tag{2}$$

For the cart M, resolving forces radially we obtain

$$N - T\cos\phi - Mg\cos\theta = -MR\dot{\theta}^2 \tag{3}$$

and resolving forces tangentially gives

$$Mg\sin\theta - T\sin\phi = MR\ddot{\theta} \tag{4}$$

We now have 4 equations with 4 unknowns. Using Mathematica the solution is

$$\theta''(t) \to \frac{g(m+2M)\sin(\theta(t)) - g\,m\sin(\theta(t)+2\phi(t)) + 2Lm\sin(\phi(t))\left(\theta'(t)+\phi'(t)\right)^2 + mR\theta'(t)^2\sin(2\phi(t))}{2R\left(m\sin^2(\phi(t))+M\right)}$$

$$\phi''(t) \to \frac{\frac{(L-\cos(\phi(t)))\left(g(m+2M)\sin(\theta(t)) - g\,m\sin(\theta(t)+2\phi(t)) + 2Lm\sin(\phi(t))\left(\theta'(t)+\phi'(t)\right)^2 + mR\theta'(t)^2\sin(2\phi(t))\right)}{2\left(m\sin^2(\phi(t))+M\right)} + g\sin(\theta(t)+\phi(t)}{L}$$

$$N \to \frac{M(m+M)\left(g\cos(\theta(t)) - R\theta'(t)^2\right) - Lm\,M\cos(\phi(t))\left(\theta'(t)+\phi'(t)\right)^2}{m\sin^2(\phi(t))+M}$$

$$T \to \frac{mM\left(\cos(\phi(t))\left(g\cos(\theta(t)) - R\theta'(t)^2\right) - L\left(\theta'(t)+\phi'(t)\right)^2\right)}{m\sin^2(\phi(t))+M}$$

Convert to first form

Let

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \theta \\ \phi \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} \to \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \ddot{\theta} \\ \ddot{\phi} \end{pmatrix}$$

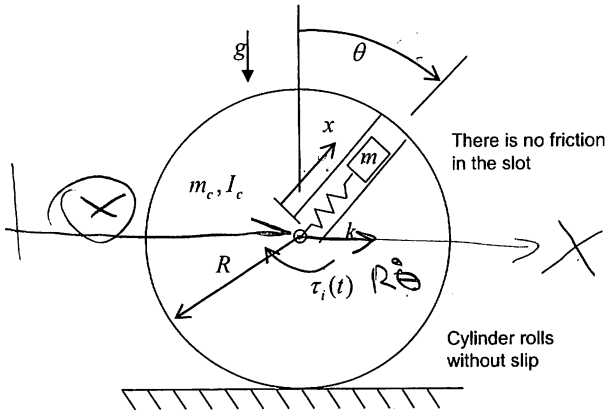
Now since the equations are decoupled, the rest follows.

2.1.4 Problem 2

Problem 2

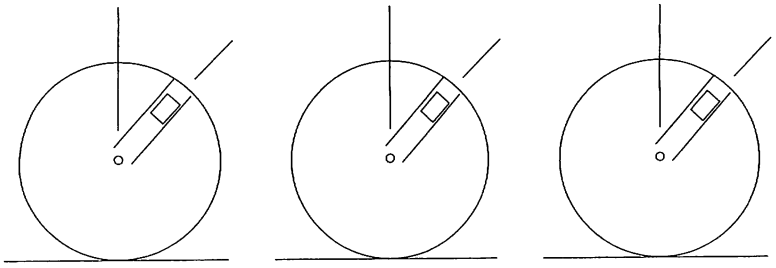
The system consists of cylinder of mass m_c, centroidal moment of inertia I_c, and radius R with a radial slot. The cylinder rolls without slip on the flat ground, its c. g. is at its center, and it is acted upon by a prescribed torque $\tau_i(t)$. Inside the slot is a mass m attached by a spring k to the center of the cylinder. The mass can slide without friction in the slot. Gravity is acting vertically downward. The system is 2 d-o-f and I have chosen the variables θ and x as indicated in the figure. We desire the equations of motion derived using direct application of Newton's laws.

when x=0 spring relaxed



Schematic of the system

A. On the figures below use arrows and symbols to show velocity, acceleration, and force components. Use only the coordinates and their derivatives for velocity and acceleration. You will have to give names to your force components.



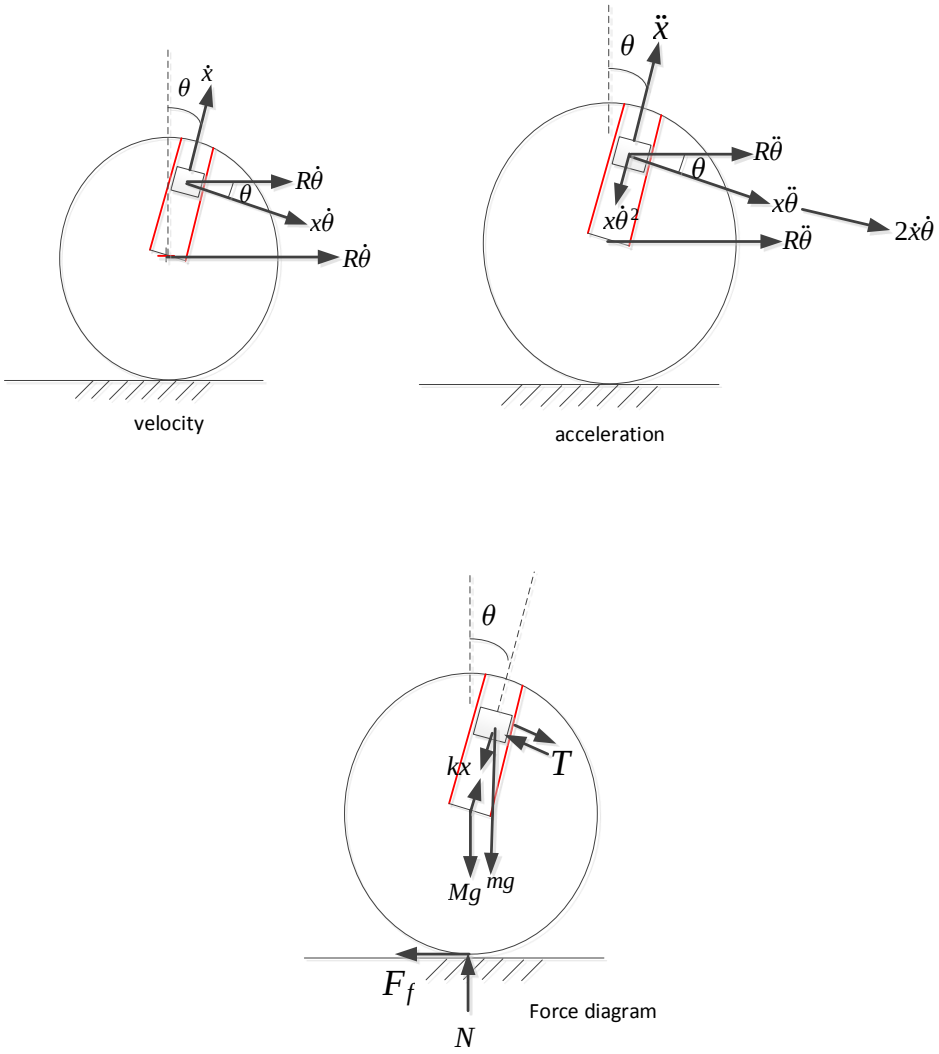
Velocity diagram | Acceleration diagram | Force diagram

B. Derive the equations of motion and show that you have sufficient equations for the unknowns.

The disk rolls with no slip, spring has no friction on the side with the disk slot. Find equations of motion.

2 DOF, using x and θ for generalized coordinates.

Velocity, acceleration and force diagrams



Solution using $F = ma$

There are 5 unknowns θ, x, T, N, F_f, hence we need 5 equations.

For mass m, resolving forces along x direction, positive upward

$$-kx - mg\cos\theta = m\left(\ddot{x} - x\dot{\theta}^2 + R\ddot{\theta}\sin\theta\right) \tag{1}$$

Resolving tangential to x direction gives

$$mg\sin\theta - T = m\left(R\ddot{\theta}\cos\theta + x\ddot{\theta} + 2\dot{x}\dot{\theta}\right) \tag{2}$$

For mass M, resolving horizontally, positive to the right results in

$$-F_f + T\cos\theta + kx\sin\theta = M\left(R\ddot{\theta}\right) \tag{3}$$

Resolving vertically, positive upwards gives

$$-Mg + N - T\sin\theta + kx\cos\theta = 0 \tag{4}$$

Applying moments around c.g. for disk results in

$$\tau + F_f R + Tx = I\ddot{\theta} \tag{5}$$

Now we have 5 equations and 5 unknowns. Solving for $\ddot{\theta}, \ddot{x}$ gives

$$\theta''(t) = \frac{\tau + gmR\cos\theta\sin\theta + (gm + Rk)\,x\sin\theta - 2m\,(R\cos\theta + x)\,\dot{x}\dot{\theta}}{I + MR^2 + mR^2\cos^2\theta + mx\,(2R\cos\theta + x)}$$

$$x''(t) = -g\cos\theta + x\left(\dot{\theta}^2 - \frac{k}{m}\right) - \frac{R\sin\theta\left(\tau + kRx\sin\theta + m\,(R\cos\theta + x)\left(g\sin\theta - 2\dot{x}\dot{\theta}\right)\right)}{I + MR^2 + mR^2\cos^2\theta + mx\,(2R\cos\theta + x)}$$

2.1.5 Convert to first form

Let

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \theta \\ x \\ \dot{\theta} \\ \dot{x} \end{pmatrix} \rightarrow \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \ddot{\theta} \\ \ddot{x} \end{pmatrix}$$

Hence

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau + gmR\cos\theta\sin\theta + (gm+Rk)x\sin\theta - 2m(R\cos\theta + x)\dot{x}\dot{\theta}}{I + MR^2 + mR^2\cos^2\theta + mx(2R\cos\theta + x)} \\ -g\cos\theta + x\left(\dot{\theta}^2 - \frac{k}{m}\right) - \frac{R\sin\theta\left(\tau + kRx\sin\theta + m(R\cos\theta + x)\left(g\sin\theta - 2\dot{x}\dot{\theta}\right)\right)}{I + MR^2 + mR^2\cos^2\theta + mx(2R\cos\theta + x)} \end{pmatrix}$$

$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau + gmR\cos x_1\sin x_1 + (gm+Rk)x_2\sin x_1 - 2m(R\cos x_1 + x_2)x_4x_3}{I + MR^2 + mR^2\cos^2 x_1 + mx(2R\cos x_1 + x_2)} \\ -g\cos x_1 + x_2\left(x_3^2 - \frac{k}{m}\right) - \frac{R\sin x_1(\tau + kRx_2\sin x_1 + m(R\cos x_1 + x_2)(g\sin x_1 - 2x_4x_3))}{I + MR^2 + mR^2\cos^2 x_1 + mx_2(2R\cos x_1 + x_2)} \end{pmatrix}$$

2.1.6 Problem 3

Problem 3

Repeat problem 2 using Lagrange equations. This is a 2 d-o-f system and the variables used in problem 2 are independent and therefore can be used for this problem. As time permits show that the motion equations from both approaches yield the same result.

The kinetic energy is

$$\begin{aligned}T &= \frac{1}{2}M\left(R\dot{\theta}\right)^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\left[\left(\dot{x} + R\dot{\theta}\sin\theta\right)^2 + \left(R\dot{\theta}\cos\theta + x\dot{\theta}\right)^2\right]\\ &= \frac{1}{2}M\left(R\dot{\theta}\right)^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\left[\dot{x}^2 + 2R\dot{x}\dot{\theta}\sin\theta + R^2\dot{\theta}^2 + x^2\dot{\theta}^2 + 2x\dot{\theta}^2R\cos\theta\right]\\ &= \frac{1}{2}M\left(R\dot{\theta}\right)^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\dot{x}^2 + mR\dot{x}\dot{\theta}\sin\theta + \frac{1}{2}m\dot{\theta}^2\left(R^2 + x^2 + 2xR\cos\theta\right)\end{aligned}$$

and the potential energy is

$$V = \frac{1}{2}kx^2 + mgx\cos\theta$$

Hence

$$\begin{aligned}L &= T - V\\ &= \frac{1}{2}M\left(R\dot{\theta}\right)^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\dot{x}^2 + mR\dot{x}\dot{\theta}\sin\theta + \frac{1}{2}m\dot{\theta}^2\left(R^2 + x^2 + 2xR\cos\theta\right) - \frac{1}{2}kx^2 - mgx\cos\theta\end{aligned}$$

For x we have

$$\begin{aligned}\frac{\partial L}{\partial \dot{x}} &= m\dot{x} + mR\dot{\theta}\sin\theta\\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= m\ddot{x} + mR\ddot{\theta}\sin\theta + mR\dot{\theta}^2\cos\theta\\ \frac{\partial L}{\partial x} &= m\dot{\theta}^2\left(x + R\cos\theta\right) - kx - mg\cos\theta\end{aligned}$$

Hence equation of motion is

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = Q_x$$

$$m\ddot{x} + mR\ddot{\theta}\sin\theta + mR\dot{\theta}^2\cos\theta - m\dot{\theta}^2\left(x + R\cos\theta\right) + kx + mg\cos\theta = 0$$

Since generalized force is zero for x. Hence from above, we have

$$\begin{aligned}\ddot{x} &= -R\ddot{\theta}\sin\theta - R\dot{\theta}^2\cos\theta + \dot{\theta}^2\left(x + R\cos\theta\right) - \frac{k}{m}x - g\cos\theta\\ &= -R\ddot{\theta}\sin\theta + \dot{\theta}^2x - \frac{k}{m}x - g\cos\theta\end{aligned} \tag{1}$$

Now for θ

$$\begin{aligned}\frac{\partial L}{\partial \dot{\theta}} &= MR^2\dot{\theta} + I\dot{\theta} + mR\dot{x}\sin\theta + m\dot{\theta}\left(R^2 + x^2 + 2xR\cos\theta\right)\\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} &= MR^2\ddot{\theta} + I\ddot{\theta} + mR\ddot{x}\sin\theta + mR\dot{x}\dot{\theta}\cos\theta + m\ddot{\theta}\left(R^2 + x^2 + 2xR\cos\theta\right) + m\dot{\theta}\left(2x\dot{x} + 2\dot{x}R\cos\theta - 2xR\dot{\theta}\sin\theta\right)\\ &= MR^2\ddot{\theta} + I\ddot{\theta} + mR\ddot{x}\sin\theta + mR\dot{x}\dot{\theta}\cos\theta + m\ddot{\theta}R^2 + m\ddot{\theta}x^2 + 2m\ddot{\theta}xR\cos\theta + 2m\dot{\theta}x\dot{x} + 2m\dot{\theta}\dot{x}R\cos\theta - 2mxR\dot{\theta}^2\sin\theta\\ &= MR^2\ddot{\theta} + I\ddot{\theta} + mR\ddot{x}\sin\theta + 3mR\dot{x}\dot{\theta}\cos\theta + m\ddot{\theta}R^2 + m\ddot{\theta}x^2 + 2m\ddot{\theta}xR\cos\theta + 2m\dot{\theta}x\dot{x} - 2mxR\dot{\theta}^2\sin\theta\\ \frac{\partial L}{\partial \theta} &= mR\dot{x}\dot{\theta}\cos\theta - m\dot{\theta}^2xR\sin\theta + mgx\sin\theta\end{aligned}$$

Hence equation of motion is

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q_\theta$$

Hence, since $Q_\theta = \tau$ then

$$MR^2\ddot{\theta} + I\ddot{\theta} + mR\ddot{x}\sin\theta + m\ddot{\theta}R^2 + m\ddot{\theta}x^2 + 2xm\ddot{\theta}R\cos\theta + 2m\dot{\theta}x\dot{x} + 2m\dot{\theta}\dot{x}R\cos\theta - m\dot{\theta}^2xR\sin\theta + mgx\sin\theta = \tau$$

or

$$\ddot{\theta}\left(I + 2xmR\cos\theta + MR^2\right) = \tau - mR\ddot{x}\sin\theta + m\ddot{\theta}R^2 + m\ddot{\theta}x^2 + 2m\dot{\theta}x\dot{x} + 2m\dot{\theta}\dot{x}R\cos\theta - m\dot{\theta}^2xR\sin\theta + mgx\sin\theta$$

$$\ddot{\theta} = \frac{\tau - mR\ddot{x}\sin\theta + m\ddot{\theta}R^2 + m\ddot{\theta}x^2 + 2m\dot{\theta}x\dot{x} + 2m\dot{\theta}\dot{x}R\cos\theta - m\dot{\theta}^2xR\sin\theta + mgx\sin\theta}{I + 2xmR\cos\theta + MR^2} \tag{2}$$

To decouple Eqs. (1) and (2), substitute (1) into (2), and then (2) into (1) to obtain

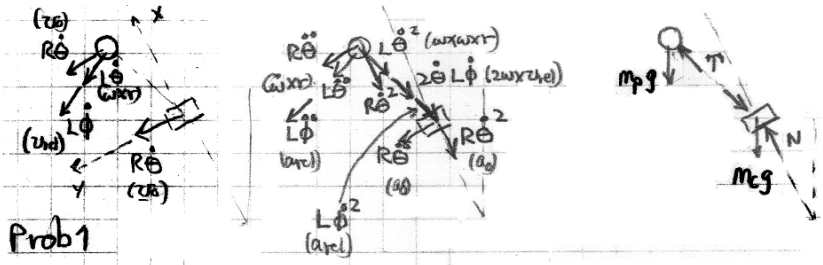
$$\theta''(t) = \frac{\tau + gmR\cos\theta\sin\theta + (gm + Rk)\,x\sin\theta - 2m\left(R\cos\theta + x\right)\dot{x}\dot{\theta}}{I + MR^2 + mR^2\cos^2\theta + mx\left(2R\cos\theta + x\right)}$$

$$x''(t) = -g\cos\theta + x\left(\dot{\theta}^2 - \frac{k}{m}\right) - \frac{R\sin\theta\left(\tau + kRx\sin\theta + m\left(R\cos\theta + x\right)\left(g\sin\theta - 2\dot{x}\dot{\theta}\right)\right)}{I + MR^2 + mR^2\cos^2\theta + mx\left(2R\cos\theta + x\right)}$$

These are the same equations found in part (2).

2.2 Key solution to midterm exam

EME 121 Midterm Exam Solution
Spring Quarter, 2011



cart: $N - T\cos\phi - m_c g\cos\theta = m_c[-R\dot\theta^2]$ ①

$m_c g\sin\theta - T\sin\phi = m_c[R\ddot\theta]$ ②

pend: $T - m_p g\cos(\theta+\phi) = m_p[R\ddot\theta\sin\phi - L\dot\theta^2 - 2L\dot\theta\dot\phi - L\dot\phi^2 - R\dot\theta^2\cos\phi]$ ③

$m_p g\sin(\theta+\phi) = m_p[R\ddot\theta\cos\phi + R\dot\theta^2\sin\phi + L\ddot\theta + L\ddot\phi]$ ④

unknowns N, T, θ, ϕ 4 Eqs. (OK)

From ② $T = \dfrac{m_c g\sin\theta}{\sin\phi} - \dfrac{m_c R\ddot\theta}{\sin\phi}$

into ③

$$\frac{m_c g\sin\theta}{\sin\phi} - \frac{m_c R\ddot\theta}{\sin\phi} - m_p g\cos(\theta+\phi) = m_p R\ddot\theta\sin\phi - m_p L\dot\theta^2 - m_p 2L\dot\theta\dot\phi - m_p L\dot\phi^2 - m_p R\dot\theta^2\cos\phi$$

$$\left[m_p R\sin\phi + \frac{m_c R}{\sin\phi}\right]\ddot\theta = m_c g\frac{\sin\theta}{\sin\phi} - m_p g\cos(\theta+\phi) + m_p L\dot\theta^2 + 2m_p L\dot\theta\dot\phi + m_p L\dot\phi^2 + m_p R\dot\theta^2\cos\phi$$

From ④

$$m_p L\ddot\phi = m_p g\sin(\theta+\phi) - m_p R\dot\theta^2 - m_p(R\cos\phi + L)\ddot\theta$$

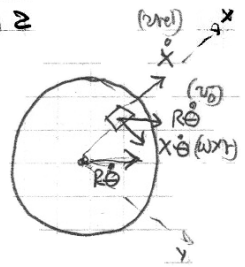
then $\dot\theta = \omega_\theta$

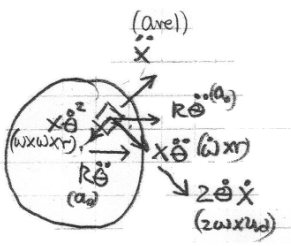
$\dot\omega_\theta = \ddot\theta = \cdots$

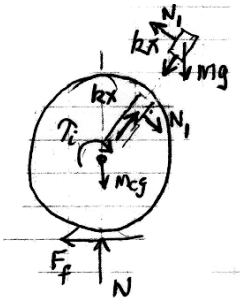
$\dot\phi = \omega_\phi$

$\dot\omega_\phi = \ddot\phi =$

Prob 2







cyl: $-F_f + kx\sin\theta + N_1\cos\theta = m_c[R\ddot{\theta}]$

$N + kx\cos\theta - N_1\sin\theta - m_c g = 0$

$F_f R + N_1 X + \tau_i = I_c\ddot{\theta}$

mass: $-kx - mg\cos\theta = m[\ddot{x} - X\dot{\theta}^2 + R\ddot{\theta}\sin\theta]$

$mg\sin\theta - N_1 = m[R\ddot{\theta}\cos\theta + X\ddot{\theta} + 2\dot{X}\dot{\theta}]$

By Lagrange $T = \frac{1}{2}m_c R^2\dot{\theta}^2 + \frac{1}{2}I_c\dot{\theta}^2 + \frac{1}{2}m[(\dot{x} + R\dot{\theta}\sin\theta)^2 + (X\dot{\theta} + R\dot{\theta}\cos\theta)^2]$

$V = mgx\cos\theta + \frac{1}{2}kx^2$

$L = \frac{1}{2}m_c R^2\dot{\theta}^2 + \frac{1}{2}I_c\dot{\theta}^2 + \frac{1}{2}m[\dot{x}^2 + R^2\dot{\theta}^2 + X^2\dot{\theta}^2 + 2R\dot{\theta}\dot{x}\sin\theta + 2RX\dot{\theta}^2\cos\theta] - mgx\cos\theta - \frac{1}{2}kx^2$

cyl: $\frac{\partial L}{\partial\dot{\theta}} = m_c R^2\dot{\theta} + I_c\dot{\theta} + mR^2\dot{\theta} + mX^2\dot{\theta} + mR\dot{x}\sin\theta + 2mRX\dot{\theta}\cos\theta$

$\frac{d}{dt}() = (I_c + m_c R^2 + mR^2)\ddot{\theta} + mX^2\ddot{\theta} + 2mX\dot{X}\dot{\theta} + mR\ddot{x}\sin\theta + mR\dot{x}\dot{\theta}\cos\theta + 2mR\dot{X}\dot{\theta}\cos\theta + 2mRX\ddot{\theta}\cos\theta - 2mRX\dot{\theta}^2\sin\theta$

$\frac{\partial L}{\partial\theta} = mR\dot{\theta}\dot{x}\cos\theta - mRX\dot{\theta}^2\sin\theta + mgX\sin\theta$

$Q_\theta = \tau_i$

$(I_c + m_c R^2 + mR^2 + mX^2)\ddot{\theta} + 2mX\dot{X}\dot{\theta} + mR\ddot{x}\sin\theta + 2mR\dot{x}\dot{\theta}\cos\theta + 2mRX\ddot{\theta}\cos\theta - mRX\dot{\theta}^2\sin\theta - mgX\sin\theta = \tau_i$

mass:

$$\frac{\partial L}{\partial \dot{x}} = m\dot{x} + mR\dot{\theta}\sin\theta$$

$$\frac{d}{dt}() = m\ddot{x} + mR\ddot{\theta}\sin\theta + mR\dot{\theta}^2\cos\theta$$

$$\frac{\partial L}{\partial x} = mx\dot{\theta}^2 + mR\dot{\theta}^2\cos\theta - mg\cos\theta - kx$$

$$Q_x = 0$$

$$m\ddot{x} + mR\ddot{\theta}\sin\theta - mx\dot{\theta}^2 + mg\cos\theta + kx = 0$$

2.3 My Final exam EME 121UC Davis, spring quarter 2011

EME 121 Engineering Applications of Dynamics
Final Exam
Professor Margolis
Take Home, open book and notes

Students are to work independently on the exam. Your signature below affirms that your work is yours alone.

Name_____

Signature_____

Problem 1 25 points _____

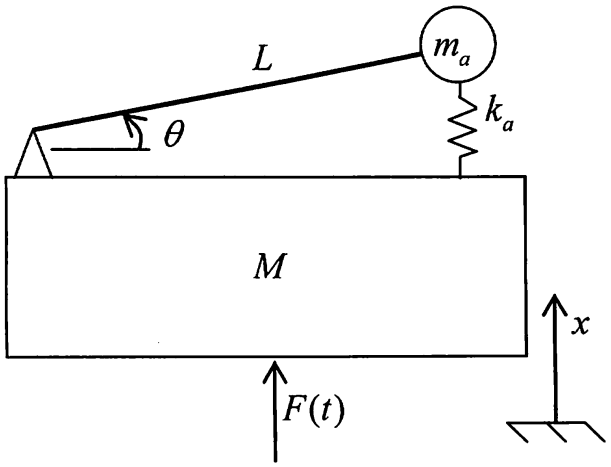
Problem 2 25 points _____

Problem 3 20 points _____

Problem 4 30 points _____

2.3.1 Problem 1

Problem 1 25 points
The figure shows a structure mass M acted on by a prescribed external force $F(t)$. On top of the mass is hinged a massless rod of length L with a mass m_a on its end. A spring k_a is between m_a and M. There is no friction in the hinge and gravity is not acting. The system is 2 d-o-f and the variables used to describe the location of this system are x to locate the structure mass and θ to locate the rod position. When $\theta = 0$ there is no force in the spring.



A. Make velocity, acceleration, and force diagrams for this system and derive the equations of motion using Newton's laws in terms of the variables and their derivatives.

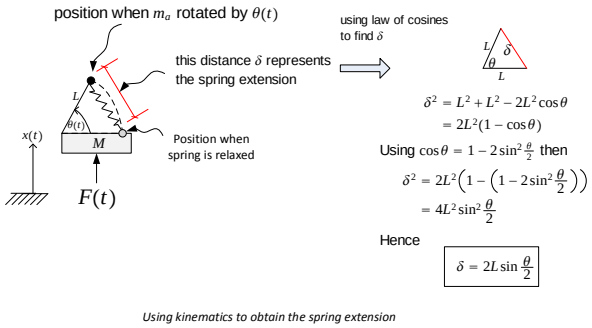
B. Cast your equations into first order form suitable for computer solution. You may discover that your equations are coupled in the second derivative terms. If you have difficulty obtaining first order form, then describe the steps you would take to obtain a final form.

The first step is to obtain an expression for the spring extension when the mass m[1] is moving and at an angle $\theta(t)$. Let $\delta(t)$ represent the current extension in spring at time t, using kinematics as shown below the following relation is found

$$\delta = 2L \sin \frac{\theta}{2}$$

The diagram below gives a snap shot of the system at time t

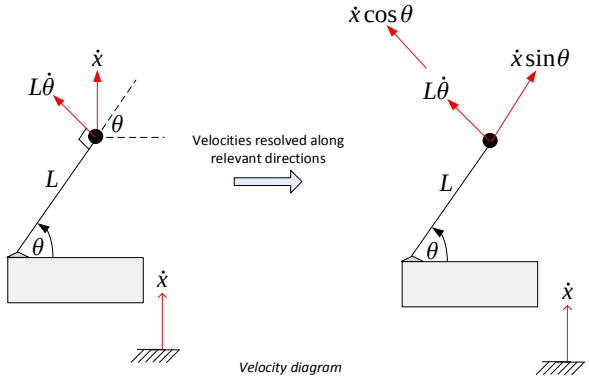
[1]For ease of typing, will use m to mean m_a



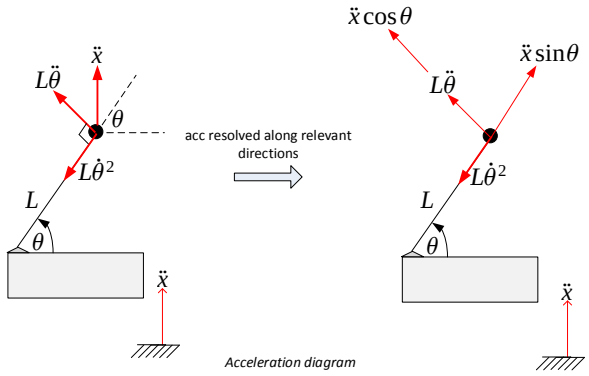
Using kinematics to obtain the spring extension

The system has 2 DOF and they are x and θ.

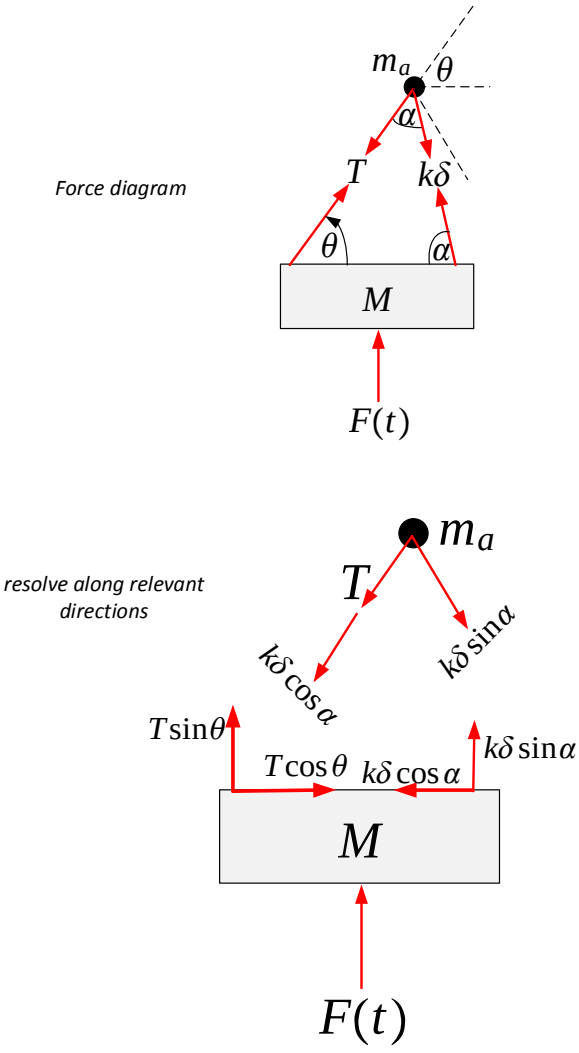
Part (A) Velocity, acceleration and force diagrams



Velocity diagram



Acceleration diagram



Generating equations of motion using $F = ma$ Using the force and acceleration diagrams, equations of motions can now be generated. There are 3 unknowns: x, θ, T, therefore 3 equations are required to solve for them.

For mass m, resolving along the radial direction results in

$$T + k\delta\cos\alpha = m\left(L\dot{\theta}^2 - \ddot{x}\sin\theta\right) \tag{1}$$

And resolving perpendicular to the radial direction gives

$$-k\delta\sin\alpha = m\left(L\ddot{\theta} + \ddot{x}\cos\theta\right) \tag{2}$$

For mass M, resolving in the vertical direction[2] gives

$$F + T\sin\theta + k\delta\sin\alpha = M\ddot{x} \tag{3}$$

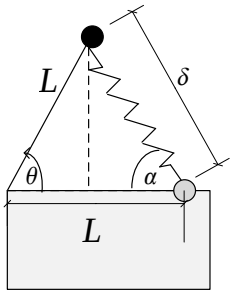
Now the $\sin\alpha$ and $\cos\alpha$ terms are expressed as functions of θ. For $\sin\alpha$, the law of sines can be used as

[2] mass M does not move in the horizontal direction, hence no equation of motion is required horizontally

follows

$$\frac{\delta}{\sin\theta} = \frac{L}{\sin\alpha}$$
$$\sin\alpha = \frac{L}{\delta}\sin\theta \tag{4}$$

An expression for $\cos\alpha$ as function of θ is now found with the help of the following diagram



$$\delta\cos\alpha + L\cos\theta = L$$
$$\cos\alpha = \frac{L - L\cos\theta}{\delta}$$
$$= \frac{L}{\delta}(1-\cos\theta)$$

Finding expression for $\cos\alpha$ as function of θ

Therefore

$$\cos\alpha = \frac{L}{\delta}(1-\cos\theta) \tag{5}$$

Substituting Eqs. (4,5) into Eqs. (1,2,3) results in new set of three equations but without the angle α explicitly appearing in the equations. Here are the 3 equations again after the above substitution. Eq. (1) becomes

$$T + k\delta\left(\frac{L}{\delta}(1-\cos\theta)\right) = m\left(L\dot{\theta}^2 - \ddot{x}\sin\theta\right)$$
$$T + kL(1-\cos\theta) = m\left(L\dot{\theta}^2 - \ddot{x}\sin\theta\right) \tag{1A}$$

And Eq. (2) becomes

$$-k\delta\left(\frac{L}{\delta}\sin\theta\right) = m\left(L\ddot{\theta} + \ddot{x}\cos\theta\right)$$
$$-kL\sin\theta = m\left(L\ddot{\theta} + \ddot{x}\cos\theta\right) \tag{2A}$$

And Eq. (3) becomes

$$F + T\sin\theta + k\delta\left(\frac{L}{\delta}\sin\theta\right) = M\ddot{x}$$
$$F + T\sin\theta + kL\sin\theta = M\ddot{x} \tag{3A}$$

Equations (1A,2A,3A) contain 3 unknowns T, x and θ. In order to obtain equation of motion in only x and θ, the unknown T is eliminated. By solving for T from Eq. (1A) and substitute the result into Eq. (3A).

Eq. (1A) gives

$$T = m\left(L\dot{\theta}^2 - \ddot{x}\sin\theta\right) - kL(1-\cos\theta)$$

Substituting the above into Eq. (3A) results in

$$\begin{aligned}
F + \overbrace{\left(m\left(L\dot{\theta}^2 - \ddot{x}\sin\theta\right) - kL\left(1-\cos\theta\right)\right)}^{T}\sin\theta + kL\sin\theta &= M\ddot{x} \\
F + mL\dot{\theta}^2\sin\theta - m\ddot{x}\sin^2\theta - kL\left(1-\cos\theta\right)\sin\theta + kL\sin\theta &= M\ddot{x} \\
F + mL\dot{\theta}^2\sin\theta - kL\sin\theta + kL\cos\theta\sin\theta + kL\sin\theta &= M\ddot{x} + m\ddot{x}\sin^2\theta \\
F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta &= \ddot{x}\left(M + m\sin^2\theta\right)
\end{aligned}$$

Hence

$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta}{M + m\sin^2\theta} \tag{6}$$

Eq. (2A) gives

$$\begin{aligned}
-kL\sin\theta &= m\left(L\ddot{\theta} + \ddot{x}\cos\theta\right) \\
mL\ddot{\theta} &= -kL\sin\theta - m\ddot{x}\cos\theta \\
\ddot{\theta} &= \frac{-kL\sin\theta - m\ddot{x}\cos\theta}{mL}
\end{aligned}$$

Hence

$$\ddot{\theta} = -\frac{k}{m}\sin\theta - \frac{\ddot{x}}{L}\cos\theta$$

This completes part(A). The equations of motion are found as given in Eq. (6) for $x(t)$ and Eq. (7) for $\theta(t)$.

The equations are seen to be coupled. In part(B), they will be decoupled and first order form generated.

Part(B)

The 2 equations of motions found in part(A) are

$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta}{M + m\sin^2\theta} \tag{1}$$

$$\ddot{\theta} = -\frac{k}{m}\sin\theta - \frac{\ddot{x}}{L}\cos\theta \tag{2}$$

Substituting (1) into (2) gives

$$\ddot{\theta} = -\frac{k}{m}\sin\theta - \frac{\cos\theta}{L}\left(\frac{F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta}{M + m\sin^2\theta}\right)$$

The equations are now decoupled. To convert to state space, 4 states variables are introduced as follows

$$\left.\begin{aligned} x_1 &= x \\ x_2 &= \theta \\ x_3 &= \dot{x} \\ x_4 &= \dot{\theta} \end{aligned}\right\} \overset{d/dt}{\rightarrow} \left.\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \ddot{x} \\ \dot{x}_4 &= \ddot{\theta} \end{aligned}\right\} \rightarrow \begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \tfrac{F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta}{M + m\sin^2\theta} \\ \dot{x}_4 &= -\tfrac{k}{m}\sin\theta - \tfrac{\cos\theta}{L}\left(\tfrac{F + mL\dot{\theta}^2\sin\theta + kL\cos\theta\sin\theta}{M + m\sin^2\theta}\right) \end{aligned}$$

Replacing all terms in the right side above by the state variables gives

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \frac{F + mLx_4^2 \sin x_2 + kL \cos x_2 \sin x_2}{M + m \sin^2 x_2} \\
\dot{x}_4 &= -\frac{k}{m} \sin x_2 - \frac{\cos x_2}{L} \left(\frac{F + mLx_4^2 \sin x_2 + kL \cos x_2 \sin x_2}{M + m \sin^2 x_2} \right)
\end{aligned}$$

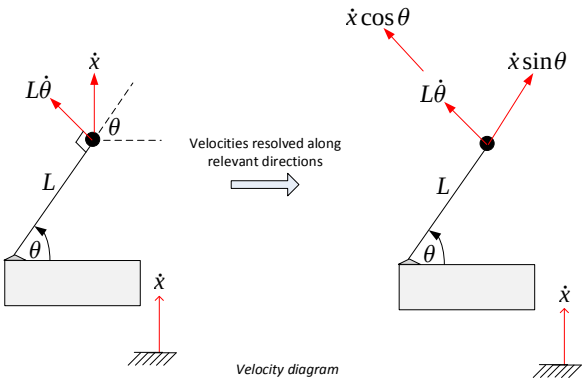
This completes problem 1.

2.3.2 Problem 2

Problem 2 25 points

Repeat **Problem 1** using Lagrange equations. Derive the equations of motion using the method of Lagrange.

Using the following velocity diagram generated in problem



The kinetic energy of the system is then given by

$$\begin{aligned}
T &= \frac{1}{2} m \left(\left[L\dot{\theta} + \dot{x} \cos\theta \right]^2 + \left[\dot{x} \sin\theta \right]^2 \right) + \frac{1}{2} M\dot{x}^2 \\
&= \frac{1}{2} m \left(L^2\dot{\theta}^2 + \dot{x}^2 + 2L\dot{\theta}\dot{x} \cos\theta \right) + \frac{1}{2} M\dot{x}^2
\end{aligned}$$

The potential energy is due to the spring extension only given by

$$V = \frac{1}{2} k\delta^2$$

But $\delta = 2L \sin\frac{\theta}{2}$ as found in problem 1, therefore

$$\begin{aligned}
V &= \frac{1}{2} k \left(2L \sin\frac{\theta}{2} \right)^2 \\
&= 2kL^2 \sin^2\frac{\theta}{2}
\end{aligned}$$

Hence the Lagrangian can be written as

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}m\left(L^2\dot{\theta}^2 + \dot{x}^2 + 2L\dot{\theta}\dot{x}\cos\theta\right) + \frac{1}{2}M\dot{x}^2 - 2kL^2\sin^2\frac{\theta}{2} \end{aligned}$$

The system has two generalized coordinates x, θ.

Starting with θ results in

$$\frac{\partial L}{\partial \dot{\theta}} = m\left(L^2\dot{\theta} + L\dot{x}\cos\theta\right)$$

And

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{1}{2}m\left(-2L\dot{\theta}\dot{x}\sin\theta\right) - 4kL^2\left(\sin\frac{\theta}{2}\right)\left(\cos\frac{\theta}{2}\right)\frac{1}{2} \\ &= -mL\dot{\theta}\dot{x}\sin\theta - 2kL^2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \end{aligned}$$

And

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} = m\left(L^2\ddot{\theta} + L\ddot{x}\cos\theta - L\dot{x}\left(\sin\theta\right)\dot{\theta}\right)$$

Therefore, the equation of motion for θ is

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q_\theta$$

But $Q_\theta = 0$ since no external force is acting to change θ and the spring force has been accounted for in the potential V, hence the above becomes

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} &= 0 \\ m\left(L^2\ddot{\theta} + L\ddot{x}\cos\theta - L\dot{x}\left(\sin\theta\right)\dot{\theta}\right) - \left(-mL\dot{\theta}\dot{x}\sin\theta - 2kL^2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\right) &= 0 \\ mL^2\ddot{\theta} + mL\ddot{x}\cos\theta - mL\dot{x}\dot{\theta}\sin\theta + mL\dot{\theta}\dot{x}\sin\theta + 2kL^2\sin\frac{\theta}{2}\cos\frac{\theta}{2} &= 0 \\ mL^2\ddot{\theta} + mL\ddot{x}\cos\theta + 2kL^2\sin\frac{\theta}{2}\cos\frac{\theta}{2} &= 0 \end{aligned}$$

Therefore

$$\begin{aligned} \ddot{\theta} &= \frac{-mL\ddot{x}\cos\theta}{mL^2} - \frac{2kL^2\sin\frac{\theta}{2}\cos\frac{\theta}{2}}{mL^2} \\ &= -\left(\frac{\ddot{x}}{L}\cos\theta + \frac{k}{m}2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\right) \end{aligned}$$

But $2\sin\frac{\theta}{2}\cos\frac{\theta}{2} = \sin\theta$[3], then then above reduces to

$$\ddot{\theta} = -\frac{\ddot{x}}{L}\cos\theta - \frac{k}{m}\sin\theta \tag{1}$$

Which is the same as was found in Eq. (1) in part(B) in the first problem when using the $F = ma$ method.

[3] Using $\sin 2A = 2\sin A\cos A$ formula

Now the equation of motion for x will be found.

$$\frac{\partial L}{\partial \dot{x}} = m\left(\dot{x} + L\dot{\theta}\cos\theta\right) + M\dot{x}$$

And

$$\frac{\partial L}{\partial x} = 0$$

But Q_x is not zero this time. $Q_x = F$ and it is positive since it adds energy to the system. This can found as follows

$$\Delta W = (\Delta x) F$$
$$Q_x = \frac{\Delta W}{\Delta x} = F$$

Continuing the derivation gives

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = m\left(\ddot{x} + L\ddot{\theta}\cos\theta - L\dot{\theta}^2\sin\theta\right) + M\ddot{x}$$

Hence the equation of motion for x is

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = Q_\theta$$
$$m\left(\ddot{x} + L\ddot{\theta}\cos\theta - L\dot{\theta}^2\sin\theta\right) + M\ddot{x} = F$$

Or

$$\ddot{x}(m+M) = F + mL\dot{\theta}^2\sin\theta - mL\ddot{\theta}\cos\theta$$
$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta - mL\ddot{\theta}\cos\theta}{m+M} \tag{2}$$

These are the equations of motion as given above in Eqs. (1,2). To decouple these equations, substituting Eq. (1) into the RHS of Eq. (2) gives

$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta - mL\overbrace{\left(-\frac{\ddot{x}}{L}\cos\theta - \frac{k}{m}\sin\theta\right)}^{\ddot{\theta}}\cos\theta}{m+M}$$

Simplifying results in

$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta + m\ddot{x}\cos^2\theta + Lk\sin\theta\cos\theta}{m+M}$$
$$\ddot{x}(m+M) - m\ddot{x}\cos^2\theta = F + mL\dot{\theta}^2\sin\theta + Lk\sin\theta\cos\theta$$
$$\ddot{x}\left(m + M - m\cos^2\theta\right) = F + mL\dot{\theta}^2\sin\theta + Lk\sin\theta\cos\theta$$
$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta + Lk\sin\theta\cos\theta}{m + M - m\cos^2\theta}$$

But $m + M - m\cos^2\theta = m\left(1 - \cos^2\theta\right) + M = m\left(\sin^2\theta\right) + M$, hence the above becomes the decoupled equation of motion for x

$$\ddot{x} = \frac{F + mL\dot{\theta}^2\sin\theta + Lk\sin\theta\cos\theta}{M + m\sin^2\theta} \tag{3}$$

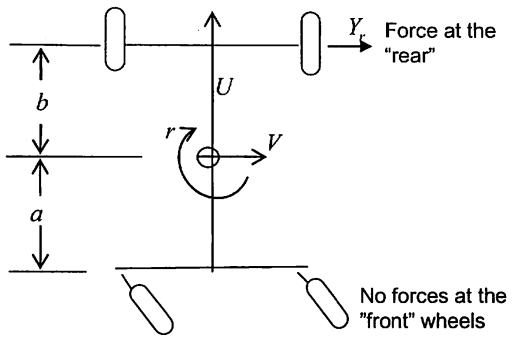
Comparing Eq. (3) above with Eq. (1) in part(B) of the first problem shows it is the same.

Therefore Eqs. (1,3) above are the equations of motion derived using Lagrangian method. They are verified to be the same equations of motion found using $F = ma$.

This completes problem 2.

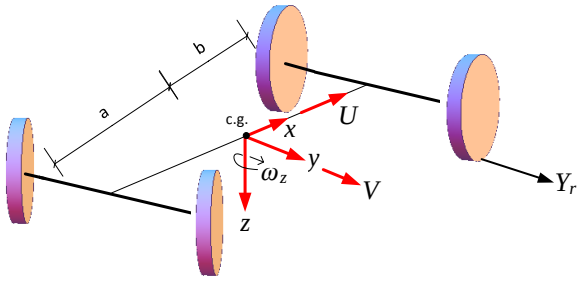
2.3.3 Problem 3

Problem 3 20 points



The figure shows the shopping cart we analyzed in class only it is going backwards. In the discussion in class, it was shown that if the forward speed was negative, the cart motion would be unstable. Your job is to redo the analysis using the new figure where U is now positive in the rearward direction. Show that the previous conclusion was correct. This amounts to redoing the eigenvalue analysis and show that this time the shopping cart is unstable for U positive. You can continue to use the "no side slip" condition at the rear as was done in the original analysis.

The following 3D diagram shows the shopping cart with body fixed coordinates system oriented such as the U is positive in the backward direction. The right hand rule was used to draw the axes with the yaw, shown as ω_z in the diagram, pointing in the positive direction as was given in the problem when viewed from top of the cart.



Showing the principle axes on the car model used for the derivation

Starting from first principles, and using the rule

$$\frac{d}{dt}\mathbf{A} = \left(\frac{d}{dt}\mathbf{A}\right)_{\text{resolved}} + \omega \times \mathbf{A}$$

The equation $F = ma$ for the body is written, where ω_z was replaced by r in the following equation

since r is the more common term.

$$\begin{aligned}
\mathbf{F} &= \frac{d}{dt}\mathbf{p} \\
&= \frac{d}{dt}(m\mathbf{v}) \\
&= m\frac{d}{dt}\mathbf{v} \\
&= m\left[\begin{pmatrix}\dot{U}\\ \dot{V}\\ \dot{Z}\end{pmatrix} + \begin{pmatrix}\omega_x\\ \omega_y\\ r\end{pmatrix} \otimes \begin{pmatrix}U\\ V\\ Z\end{pmatrix}\right] \\
&= m\left[\begin{pmatrix}\dot{U}\\ \dot{V}\\ \dot{Z}\end{pmatrix} + \det\begin{vmatrix}\mathbf{i} & \mathbf{j} & \mathbf{k}\\ \omega_x & \omega_y & r\\ U & V & Z\end{vmatrix}\right] \\
&= m\left[\begin{pmatrix}\dot{U}\\ \dot{V}\\ \dot{Z}\end{pmatrix} + \begin{pmatrix}\omega_y Z - rV\\ -(\omega_x V - rU)\\ \omega_x V - \omega_y U\end{pmatrix}\right]
\end{aligned}$$

The only angular velocity present is r and $\dot{Z} = 0$ since there is no motion in the z direction, therefore the above simplifies to

$$\begin{pmatrix}F_x\\ F_y\end{pmatrix} = m\begin{pmatrix}\dot{U} - rV\\ \dot{V} + rU\end{pmatrix}$$

And since U is constant U_0 and since there is no force in the x direction, the above simplifies to just the following equation

$$F_y = m\left(\dot{V} + rU_0\right)$$

Where $F_y = Y_r$ hence

$$Y_r = m\left(\dot{V} + rU_0\right) \tag{1}$$

Now the torque equation $\tau = I\omega$ on the rigid body is applied. Using the principle axes, the moment of inertia matrix is

$$I = \begin{pmatrix}I_1 & 0 & 0\\ 0 & I_2 & 0\\ 0 & 0 & I_3\end{pmatrix} \equiv \begin{pmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{pmatrix}$$

Hence

$$\tau = \frac{d}{dt}\left[\begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{pmatrix}\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}\right] + \begin{pmatrix} \omega_x \\ \omega_y \\ r \end{pmatrix} \times \begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{pmatrix}\begin{pmatrix} \omega_x \\ \omega_y \\ r \end{pmatrix}$$

$$\begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{pmatrix}\begin{pmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{r} \end{pmatrix} + \begin{pmatrix} \omega_x \\ \omega_y \\ r \end{pmatrix} \times \begin{pmatrix} I_1\omega_x \\ I_2\omega_y \\ I_3 r \end{pmatrix}$$

$$= \begin{pmatrix} I_1\dot{\omega}_x \\ I_2\dot{\omega}_y \\ I_3\dot{r} \end{pmatrix} + \det\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \omega_x & \omega_y & r \\ I_1\omega_x & I_2\omega_y & I_3 r \end{vmatrix}$$

$$= \begin{pmatrix} I_1\dot{\omega}_x \\ I_2\dot{\omega}_y \\ I_3\dot{r} \end{pmatrix} + \begin{pmatrix} \omega_y (I_3 r) - r\left(I_2\omega_y\right) \\ -\omega_x (I_3 r) + r (I_1\omega_x) \\ \omega_x \left(I_2\omega_y\right) - \omega_y (I_1\omega_x) \end{pmatrix}$$

$$= \begin{pmatrix} I_1\dot{\omega}_x \\ I_2\dot{\omega}_y \\ I_3\dot{r} \end{pmatrix} + \begin{pmatrix} \omega_y r (I_3 - I_2) \\ \omega_x r (I_1 - I_3) \\ \omega_x\omega_y (I_2 - I_1) \end{pmatrix}$$

Since $\omega_y = \omega_x = 0$ the above simplifies to one torque equation

$$\tau_z = I_3\dot{r}$$

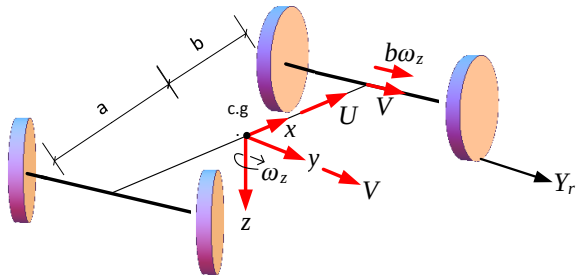
Where $\tau_z = bY_r$, hence the above becomes

$$bY_r = I_3\dot{r} \tag{2}$$

Substituting Eq.(1) into (2) results in

$$bm\left(\dot{V} + rU_0\right) = I_3\dot{r} \tag{3}$$

Now the no-slip condition is applied to the rear tire. Transferring the center of mass velocity to the back axial is illustrated in the following diagram



From the above it can be seen that the no-slip condition leads to the following result

$$V + br = 0$$

Or

$$V = -br \tag{4}$$

Substituting Eq. (4) back into Eq. (3) results in

$$bm\left(-b\dot{r}+rU_0\right)=I_3\dot{r}$$
$$-mb^2\dot{r}+rbmU_0=I_3\dot{r}$$

Hence

$$\dot{r}\left(I_3+mb^2\right)-\left(bmU_0\right)r=0 \tag{5}$$

This is the equation of motion for r. Assuming the solution is $r=\bar{r}e^{st}$ then the above becomes

$$\bar{r}se^{st}\left(I_3+mb^2\right)-\left(bmU_0\right)\bar{r}e^{st}=0$$

For non-trivial solution, $\bar{r}\neq 0$ and $e^{st}\neq 0$ hence dividing by $\bar{r}e^{st}$ gives

$$s\left(I_3+mb^2\right)-\left(bmU_0\right)=0$$

Therefore

$$s=\frac{bmU_0}{I_3+mb^2}$$

The quantity in the denominator is positive always. Hence, U_0 is assumed to be positive going backward, then this implies that s above is positive as all other quantities shown are positive.

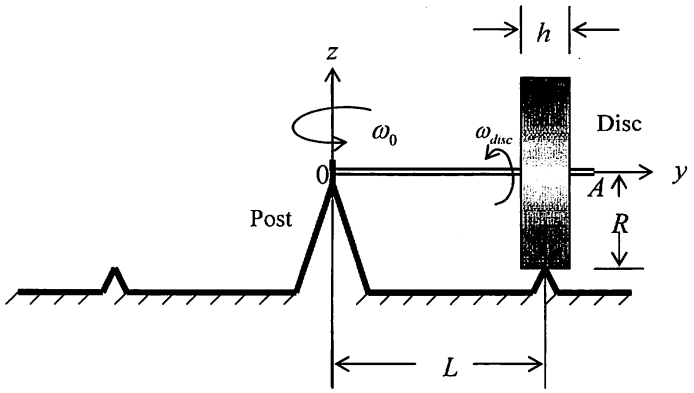
Therefore it is concluded that the cart is unstable as the eigenvalue is positive.

This result agrees with the result given in class. Changing the axes orientation did not change the final conclusion, which is the cart will be unstable when moving in the rear direction.

This completes problem 3.

2.3.4 Problem 4

Problem 4 30 points



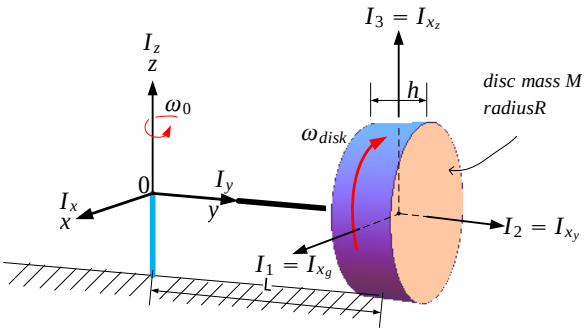
The disc with uniform mass density and total mass M rolls without slip on a horizontal circle of radius L. The axle from point 0 to A through the center of the disc has negligible mass and rotates with angular speed ω_0 about the post. Note that the coordinate frame $0xyz$ (the x-axis is pointing out of the page at the instant shown)rotates with the axle around the post but does not participate in the spin of the disc around the axle. However, because of symmetry, the frame $0xyz$ can be considered to be a principle axis coordinate system for the disc with principal moments of inertia about point 0 of I_x, I_y and I_z where $I_x = I_z$ due to the symmetry.

a. If $I_{x_g}, I_{y_g}, I_{z_g}$ are the principal moments of inertia of the disk with respect to axes at the c. g., derive or write down these moments of inertia in terms of the system parameters M, L, R or anything else you might need. What are the principal moments of inertia with respect to the axes through point 0?

b. Using the no-slip condition, find the angular velocity *of the disc* $\boldsymbol{\omega}_{disc}$ as it depends on the angular velocity ω_0.

c. Find the kinetic energy of the disc in terms of ω_0 and the system parameters.

Part(a)



The above diagram shows the principal axes for the disk through its c.g. These axes are fixed on the disk and rotate with it. The principal moments of inertia matrix of the disk around its c.g. can be found by integration. They are shown here from tables

$$\left[I_{cg}\right] = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{12}M\left(3R^2 + h^2\right) & 0 & 0 \\ 0 & \frac{MR^2}{2} & 0 \\ 0 & 0 & \frac{1}{12}M\left(3R^2 + h^2\right) \end{bmatrix}$$

To find the principal moments of inertia matrix around axes through 0, the parallel axes theorem is used. Hence

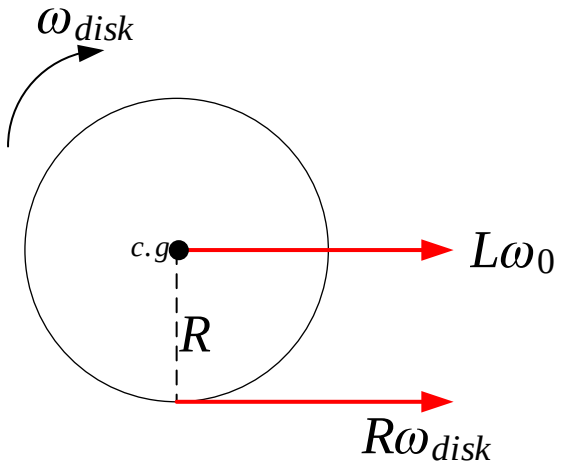
$$\begin{aligned} [I_0] &= \left[I_{cg}\right] + M\begin{bmatrix} L^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & L^2 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{12}M\left(3R^2 + h^2\right) & 0 & 0 \\ 0 & \frac{MR^2}{2} & 0 \\ 0 & 0 & \frac{1}{12}M\left(3R^2 + h^2\right) \end{bmatrix} + \begin{bmatrix} ML^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & ML^2 \end{bmatrix} \end{aligned}$$

Therefore

$$\begin{aligned} [I_0] &= \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \\ &= \begin{bmatrix} M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right) & 0 & 0 \\ 0 & \frac{MR^2}{2} & 0 \\ 0 & 0 & M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right) \end{bmatrix} \end{aligned}$$

Part (b)

Viewing the disk from the front, looking at it when standing in front of the $0y$ axis facing the disk, the following velocity components can be seen



The velocity $L\omega_0$ is the instantaneous linear velocity of the c.g. of the disk due to the disk rotation around the post. The term $R\omega_{disk}$ is the instantaneous linear velocity of the point where the disk touching the ground.

For no-slip these two velocities must be equal which means

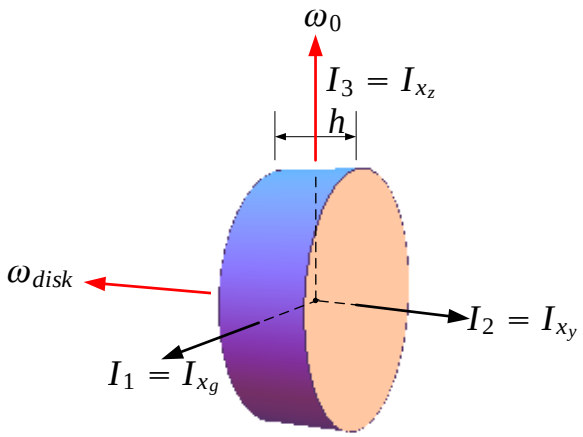
$$R\omega_{disk} = L\omega_0$$

Hence

$$\omega_{disk} = \frac{L}{R}\omega_0$$

Part (c)

The disk has kinetic energy due to the spins around its principal axes and due to orbital translation around the base post located at 0.



Therefore

$$\begin{aligned} T &= \frac{1}{2}M\left(L\omega_0\right)^2 + \frac{1}{2}I_2\omega_{disk}^2 + \frac{1}{2}I_3\omega_0^2 \\ &= \frac{1}{2}ML^2\omega_0^2 + \frac{1}{2}\frac{MR^2}{2}\omega_{disk}^2 + \frac{1}{2}M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0^2 \end{aligned}$$

Using the result found from (b) that $\omega_{disk} = \frac{L}{R}\omega_0$ the above reduces to

$$\begin{aligned} T &= \frac{1}{2}ML^2\omega_0^2 + \frac{MR^2}{4}\frac{L^2}{R^2}\omega_0^2 + \frac{1}{2}M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0^2 \\ &= \frac{h^2M\omega_0^2}{24} + ML^2\omega_0^2 + \frac{1}{8}MR^2\omega_0^2 + \frac{L^2MR^2\omega_0^2}{4R^2} \end{aligned}$$

Therefore

$$T = \frac{M\omega_0^2}{24}\left(h^2 + 30L^2 + 3R^2\right)$$

Part (d)

The angular momentum of the disk around 0 is

$$\mathbf{H}_0 = I_0\omega$$

where now the moment of inertia tensor used in the one relative to 0 and not relative to the $c.g.$ of disk. Angular velocities do not change. Only the moment of inertia tensor is changed. This works since the disk has one point that is fixed in space at 0 as it rotates. This is due to the axial that extends from its center to point 0 to the disk, which is considered part of the disk, but contribute no inertia since it is assumed to be massless.

Therefore, the angular momentum around 0 can be written as

$$\begin{aligned} \mathbf{H}_0 &= \begin{bmatrix} M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right) & 0 & 0 \\ 0 & \frac{MR^2}{2} & 0 \\ 0 & 0 & M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right) \end{bmatrix} \begin{bmatrix} 0 \\ -\omega_{disk} \\ \omega_0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ -\frac{MR^2}{2}\omega_{disk} \\ M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0 \end{bmatrix} \end{aligned}$$

Hence

$$\mathbf{H}_0 = 0\,\mathbf{i} - \left(\frac{MR^2}{2}\omega_{disk}\right)\mathbf{j} + M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0\mathbf{k}$$

Therefore it magnitude is

$$\left|\mathbf{H}_0\right| = \sqrt{\left(\frac{MR^2}{2}\omega_{disk}\right)^2 + \left(M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0\right)^2}$$

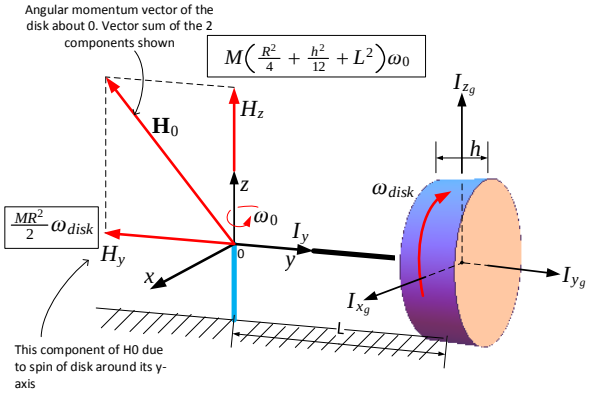
Using the no-slip condition $\omega_{disk} = \frac{L}{R}\omega_0$ the above simplifies to

$$|\mathbf{H}_0| = \sqrt{\left(\frac{MR^2}{2}\frac{L}{R}\omega_0\right)^2 + \left(M\left(\frac{R^2}{4}+\frac{h^2}{12}+L^2\right)\omega_0\right)^2}$$

$$|\mathbf{H}_0| = \sqrt{\frac{M^2R^2L^2\omega_0^2}{4} + M^2\left(\frac{R^2}{4}+\frac{h^2}{12}+L^2\right)^2\omega_0^2}$$

$$= M\omega_0\sqrt{\frac{R^2L^2}{4} + \left(\frac{R^2}{4}+\frac{h^2}{12}+L^2\right)^2}$$

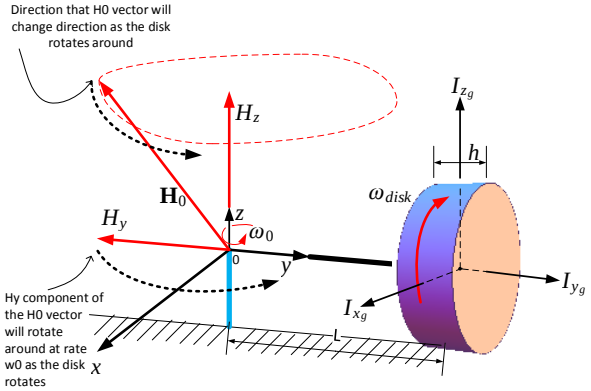
The direction of $\mathbf{H}_0$ is given in the diagram below



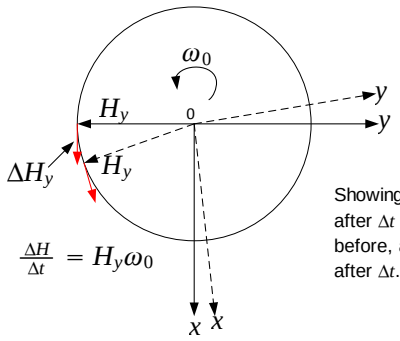
Part(e)

A constant ω_0 implies that the component H_z of the angular momentum is constant (in both magnitude and direction). Therefore the H_y component (due to the disk spin around y axis) is the only component that will be changing. The change comes due to change in direction and not in magnitude as the disk rotates around the base.

The tip of the $\mathbf{H}_0$ vector therefore travels along a circle above 0 at a distance given by magnitude of the constant H_z component. The vector $\mathbf{H}_0$ will be rotating at the constant rate of ω_0 as illustrated in the following diagram



Looking at H_y component from above helps to determine the rate of change of angular momentum



Showing H_y changing direction after Δt time the solid lines are before, and the dashed lines after Δt. Hence $\frac{\Delta H}{\Delta t} = H_y\omega_0$

The above diagram shows the instantaneous rate of change of the vector $\mathbf{H}_0$. The vector $\Delta\mathbf{H}$ is tangent to the circle at the tip of the $\mathbf{H}_0$ vector. Hence $\Delta\mathbf{H}$ points towards the x axis.

Since $\mathbf{H}_0$ rotates at rate ω_0 this results in

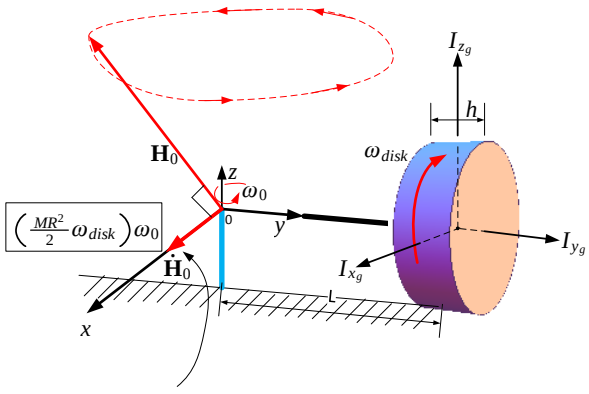
$$\frac{\Delta \mathbf{H}_0}{\Delta t} = H_y\omega_0\mathbf{i}$$

Where $H_y = \frac{MR^2}{2}\omega_{disk}$ found earlier. (The minus sign has been accounted for by drawing the component in the negative direction above).

As can be seen from the diagram, the rate of change of the angular momentum (which is also a vector), always points in the positive x direction. Therefore, in vector form, $\mathbf{H}_0$ can be written as

$$\mathbf{H}_0 = \begin{bmatrix} H_y\omega_0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \left(\frac{MR^2}{2}\omega_{disk}\right)\omega_0 \\ 0 \\ 0 \end{bmatrix} = \left(\frac{MR^2}{2}\omega_{disk}\right)\omega_0\,\mathbf{i} + 0\,\mathbf{j} + 0\mathbf{k}$$

The following diagram shows the direction of $\mathbf{H}_0$ in the original diagram to help illustrate it better



$\dot{\mathbf{H}}_0$ points along the x axis as it rotates around. Direction of $\dot{\mathbf{H}}_0$ is the same direction as the torque which causes it. Therefore, there must be a torque $\boldsymbol{\tau}$ vector pointing in the x direction.

Using the no-slip condition $\omega_{disk} = \frac{L}{R}\omega_0$ in the above simplifies $\mathbf{H}_0$ resulting in

$$\mathbf{H}_0 = \begin{bmatrix} \left(\frac{MR^2}{2}\omega_{disk}\right)\omega_0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \left(\frac{MR^2}{2}\frac{L}{R}\omega_0\right)\omega_0 \\ 0 \\ 0 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix}$$

Therefore, since rate of change of angular momentum is not zero and has magnitude of $\frac{MR}{2}L\omega_0^2$ and points in the positive x direction, and since

$$\frac{d}{dt}\mathbf{H}_0 = \tau_0$$

Then there must be a torque τ_0 acting around the x axis. In the next part this torque vector is verified and it cause outlined.

Part(f)

There exist a centripetal force due to rotation of the disk that points towards the 0. This force vector has a moment arm which is the length R which is the height of the c.g. of the desk above the ground. Therefore, this produces a torque around the x axis as shown below.

Using the right hand rule, the torque vector can be seen to point in the x direction, which is the direction that the rate of change of angular momentum points to as would be expected.

The amount of the centripetal force is

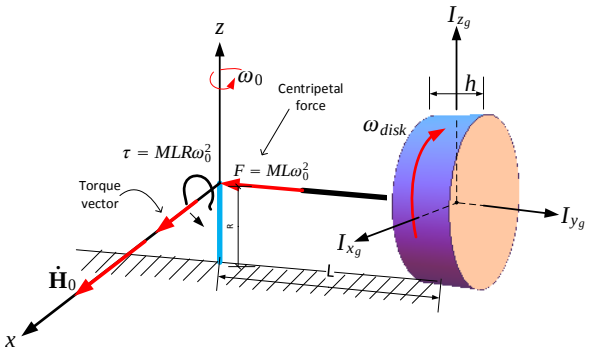
$$F = Ma_c$$

Where a_c is the disk acceleration towards the center 0 which is $L\omega_0^2$. Therefore

$$F = ML\omega_0^2$$

Hence the torque magnitude is given by

$$\tau = MLR\omega_0^2$$



To verify that the above torque is equal to $\frac{d}{dt}\mathbf{H}_0$, then applying the definition of $\frac{d}{dt}\mathbf{H}_0$ and accounting for the rotation around 0 results in

$$\begin{aligned}
\tau_0 &= \frac{d}{dt}\mathbf{H}_0 \\
&= \left(\frac{d}{dt}\mathbf{H}_0\right)_{0xyz} + \omega \times \mathbf{H}_0 \\
&= \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix} + \det \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & \omega_0 \\ 0 & -\frac{MR^2}{2}\omega_{disk} & M\left(\frac{R^2}{4} + \frac{h^2}{12} + L^2\right)\omega_0 \end{vmatrix} \\
&= \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \omega_0 \frac{MR^2}{2}\omega_{disk} \\ 0 \\ 0 \end{bmatrix}
\end{aligned}$$

Using the no-slip condition $\omega_{disk} = \frac{L}{R}\omega_0$, the above becomes

$$\begin{aligned}
\tau_0 &= \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \omega_0 \frac{MR^2}{2}\frac{L}{R}\omega_0 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{MR}{2}L\omega_0^2 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} MRL\omega_0^2 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}$$

Or in vector form

$$\tau_0 = MRL\omega_0^2\ \mathbf{i}+0\mathbf{j}+0\mathbf{k}$$

And the magnitude of the torque is

$$\tau = MRL\omega_0^2$$

Which is the same as was found above. This completes problem 4.

Chapter 3

my HW solutions

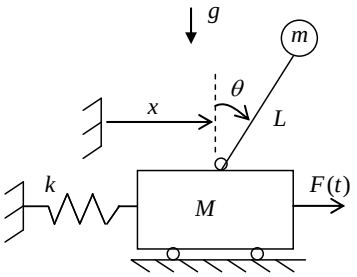
This PDF contains scan of my solution to the main HW problems from the book. About 30 or so.

3.1 First HW set

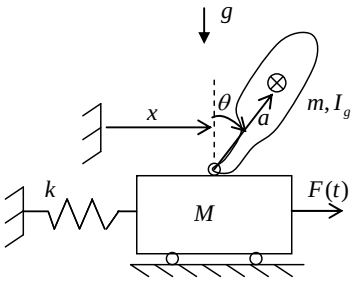
Solve number of mechanical problems, generate velocity, acceleration and force diagrams, and the equations of motions. Add solution using Lagrangian as well. 5 problems

EME 121 Practice problems

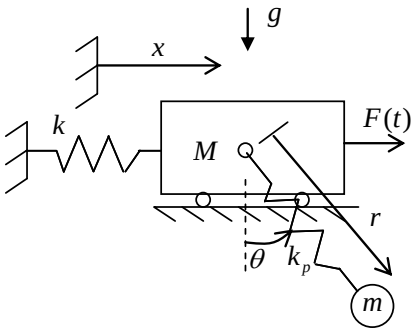
Make velocity, acceleration and force diagrams for these problems and derive the equations of motion using Newton's laws. Present results with highest order derivatives on the left and the lower order terms on the right. Reduce to first order form.

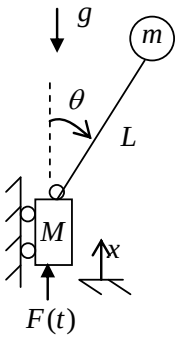


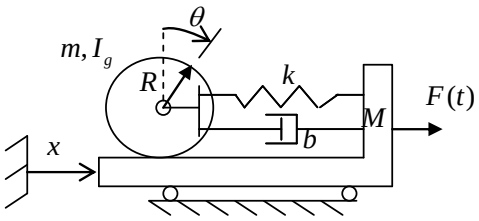
When x=0, spring is relaxed



When x=0, spring is relaxed







Cylinder rolls without slip on cart

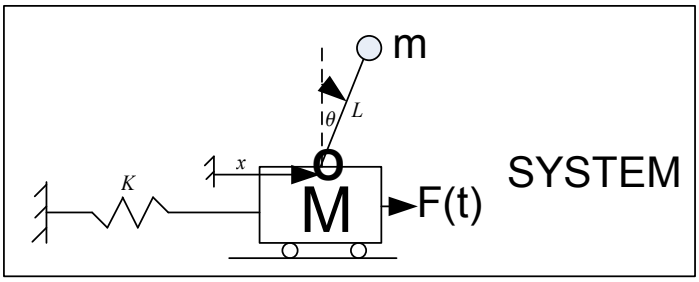
Spring/damper attached to center of cylinder

When θ is zero, spring is relaxed

3.1.1 problem 1

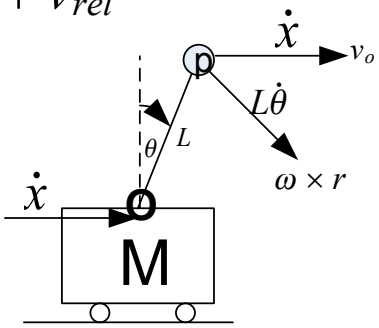
Velocity, acceleration and force diagrams

Generalized coordinates are: x, θ

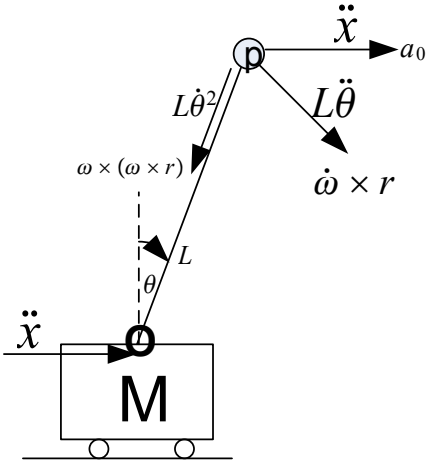


$$v_p = v_o + \omega \times r + v_{rel}$$

ω points into the paper

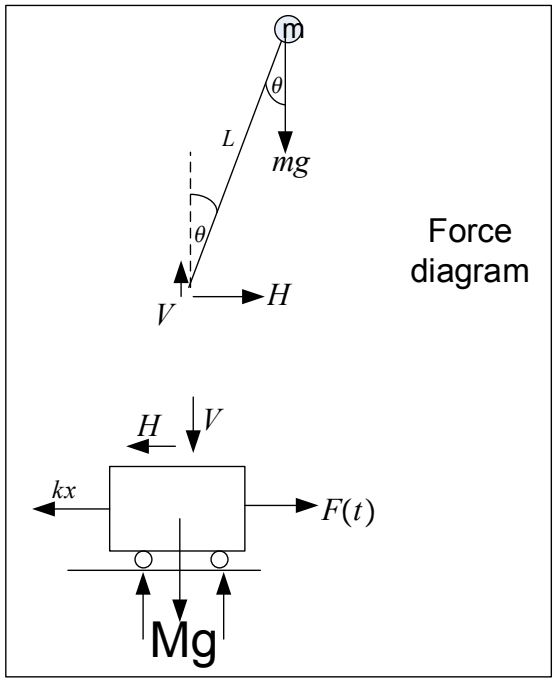


VELOCITY



Accelaration

$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$



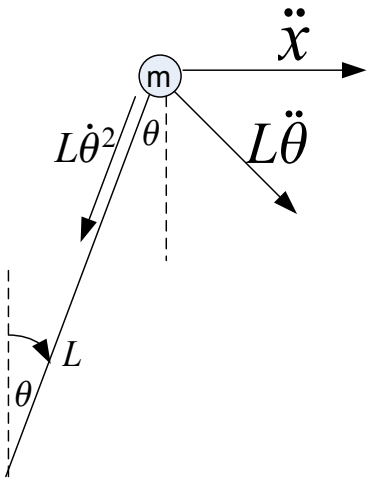
Deriving equations of motion using direct force method

There are 4 unknowns in the system, H, V, x'', θ''. Therefore, we need 4 equations.

Starting with equation of motion for mass M

$$\overrightarrow{\sum} F_x = Mx''$$
$$F(t) - kx - H = Mx'' \tag{1}$$

For mass m



$$\begin{aligned}\overrightarrow{\sum} F_x &= m\left(x'' + L\theta'' \cos\theta - L(\theta')^2 \sin\theta\right) \\ H &= m\left(x'' + L\theta'' \cos\theta - L(\theta')^2 \sin\theta\right) \end{aligned} \tag{2}$$

$$\begin{aligned}\uparrow \sum F_y &= m\left(-L\theta'' \sin\theta - L(\theta')^2 \cos\theta\right) \\ V - mg &= -m\left(L\theta'' \sin\theta + L(\theta')^2 \cos\theta\right) \end{aligned} \tag{3}$$

Take moments around mass m, anti-clockwise is positive, hence

$$\begin{aligned} HL\cos\theta - VL\sin\theta &= 0 \\ V &= H\frac{\cos\theta}{\sin\theta} \end{aligned}$$

Substituting the above in (3) gives

$$H\frac{\cos\theta}{\sin\theta} - mg = -m\left(L\theta'' \sin\theta + L\left(\theta'\right)^2 \cos\theta\right)$$

Now, substituting H from (2) into the above gives

$$\begin{aligned} \frac{m\left(x'' + L\theta'' \cos\theta - L\left(\theta'\right)^2 \sin\theta\right)\cos\theta}{\sin\theta} - mg &= -m\left(L\theta'' \sin\theta + L\left(\theta'\right)^2 \cos\theta\right) \\ \left(x'' + L\theta'' \cos\theta - L\left(\theta'\right)^2 \sin\theta\right)\cos\theta - g\sin\theta &= -L\theta'' \sin^2\theta - L\left(\theta'\right)^2 \cos\theta \sin\theta \\ L\theta'' \cos^2\theta + L\theta'' \sin^2\theta &= g\sin\theta - x'' \cos\theta \\ \theta'' &= \frac{g\sin\theta - x''\cos\theta}{L} \end{aligned}$$

Now, substituting H into (1) gives

$$\begin{aligned} F(t) - kx - H &= Mx'' \\ F(t) - kx - m\left(x'' + L\theta'' \cos\theta - L\left(\theta'\right)^2 \sin\theta\right) &= Mx'' \\ x''(M+m) &= F(t) - kx - mL\theta'' \cos\theta + mL\left(\theta'\right)^2 \sin\theta \\ x'' &= \frac{F(t) - kx - mL\theta'' \cos\theta + mL\left(\theta'\right)^2 \sin\theta}{M+m} \end{aligned}$$

Therefore, the final EQM are

$$x'' = \frac{F(t) - kx - mL\theta'' \cos\theta + mL\left(\theta'\right)^2 \sin\theta}{M+m} \tag{4}$$

$$\theta'' = \frac{g\sin\theta - x''\cos\theta}{L} \tag{5}$$

Decouple the ODE's

Substitute Eq. (5) into Eq. (4) gives

$$x'' = \frac{F(t) - kx - mL\left[\frac{g\sin\theta - x''\cos\theta}{L}\right]\cos\theta + mL(\theta')^2\sin\theta}{M+m}$$
$$= \frac{F(t) - kx - mg\sin\theta\cos\theta + mx''\cos^2\theta + mL(\theta')^2\sin\theta}{M+m}$$
$$x''(M+m) - mx''\cos^2\theta = F(t) - kx - mg\sin\theta\cos\theta + mL(\theta')^2\sin\theta$$
$$x'' = \frac{F(t) - kx - mg\sin\theta\cos\theta + mL(\theta')^2\sin\theta}{(M+m-m\cos^2\theta)} \tag{4A}$$

Also, we can solve for θ'' by Substituting Eq. (4) into (5)

$$\theta'' = \frac{g\sin\theta - \left(\frac{F(t) - kx - mL\theta''\cos\theta + mL(\theta')^2\sin\theta}{M+m}\right)\cos\theta}{L}$$
$$L\theta''(M+m) = g\sin\theta(M+m) - \left(F(t) - kx - mL\theta''\cos\theta + mL(\theta')^2\sin\theta\right)\cos\theta$$
$$L\theta''(M+m) = g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta + mL\theta''\cos^2\theta - mL(\theta')^2\sin\theta\cos\theta$$
$$L\theta''(M+m) - mL\theta''\cos^2\theta = g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta - mL(\theta')^2\sin\theta\cos\theta$$
$$\theta'' = \frac{g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta - mL(\theta')^2\sin\theta\cos\theta}{L(M+m-m\cos^2\theta)} \tag{5A}$$

Eqs. (4A) and (5A) is what will be used to convert the system to state space since they are in decoupled form.

Convert to state space form

Using the decoupled ODE's above, Eqs (4A) and (5A), and introducing 4 state variables $x_1, x_2, x_3 x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t) - kx - mg\sin\theta\cos\theta + mL(\theta')^2\sin\theta}{(M+m-m\cos^2\theta)} \\ \frac{g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta - mL(\theta')^2\sin\theta\cos\theta}{L(M+m-m\cos^2\theta)} \end{pmatrix}$$
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t) - kx_1 - mg\sin x_2\cos x_2 + mLx_4^2\sin x_1}{(M+m-m\cos^2 x_2)} \\ \frac{g\sin x_2(M+m) - F(t)\cos x_2 + kx_1\cos x_2 - mLx_4^2\sin x_2\cos x_2}{L(M+m-m\cos^2 x_2)} \end{pmatrix}$$

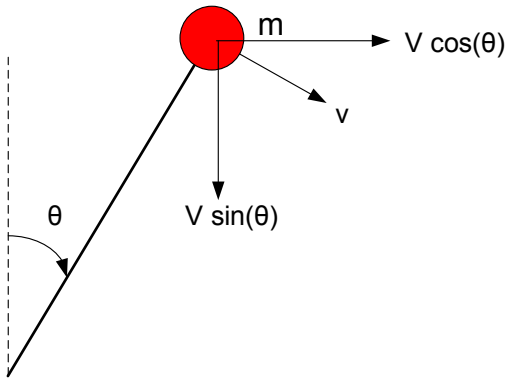
Deriving equations of motion using Lagrangian method (for verification)

To verify the solution above, try using Lagrangian method. Let L be the Lagrangian, and let T be the kinetic energy of the system, and U the potential energy.

$$L = T - U$$

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}mv^2$$

Where v is the speed of the mass m which is



$$\begin{aligned} v^2 &= (\dot{x} + v_x)^2 + v_y^2 \\ &= (\dot{x} + v\cos\theta)^2 + (v\sin\theta)^2 \\ &= \left(\dot{x} + L\dot{\theta}\cos\theta\right)^2 + \left(L\dot{\theta}\sin\theta\right)^2 \\ &= \dot{x}^2 + \left(L\dot{\theta}\right)^2\cos^2\theta + 2\dot{x}L\dot{\theta}\cos\theta + \left(L\dot{\theta}\right)^2\sin^2\theta \\ &= \dot{x}^2 + \left(L\dot{\theta}\right)^2 + 2\dot{x}L\dot{\theta}\cos\theta \end{aligned}$$

Hence

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(L\dot{\theta}\right)^2 + 2\dot{x}L\dot{\theta}\cos\theta\right)$$

For the potential energy, the mass M has

$$U_M = \frac{1}{2}kx^2$$

and the mass m is losing PE, hence, assuming that the zero potential energy is at the ground level

$$U_m = mgL\cos\theta$$

Therefore, L becomes

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(L\dot{\theta}\right)^2 + 2\dot{x}L\dot{\theta}\cos\theta\right) - \left(\frac{1}{2}kx^2 + mgL\cos\theta\right)$$

To obtain equations of motion, evaluate

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F \tag{1}$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{2}$$

For M we obtain

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} + mL\dot{\theta}\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = M\ddot{x} + m\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x} = -kx$$

Hence, for the mass M, the equation of motion is

$$M\ddot{x} + m\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta + kx = F$$

$$\ddot{x} = \frac{F - mL\ddot{\theta}\cos\theta + mL\dot{\theta}^2\sin\theta - kx}{M+m} \tag{4C}$$

To find EQM for mass m, evaluate $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$

$$\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} + m\dot{x}L\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = mL^2\ddot{\theta} + m\ddot{x}L\cos\theta - m\dot{x}L\dot{\theta}\sin\theta$$

$$\frac{\partial L}{\partial \theta} = -m\dot{x}L\dot{\theta}\sin\theta + mgL\sin\theta$$

Hence EQM is

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$$

$$mL^2\ddot{\theta} + m\ddot{x}L\cos\theta - m\dot{x}L\dot{\theta}\sin\theta - \left(-m\dot{x}L\dot{\theta}\sin\theta + mgL\sin\theta\right) = 0$$

$$L\ddot{\theta} + \ddot{x}\cos\theta - \dot{x}\dot{\theta}\sin\theta + \dot{x}\dot{\theta}\sin\theta - g\sin\theta = 0$$

$$L\ddot{\theta} + \ddot{x}\cos\theta - g\sin\theta = 0$$

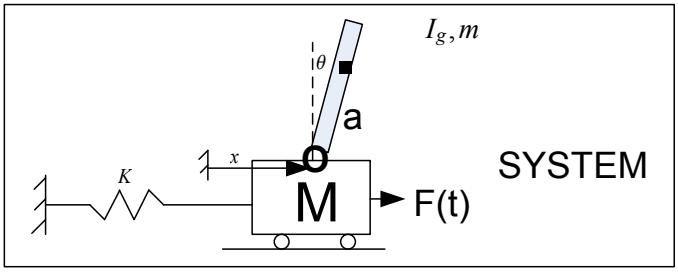
$$\ddot{\theta} = \frac{g\sin\theta - \ddot{x}\cos\theta}{L} \tag{5C}$$

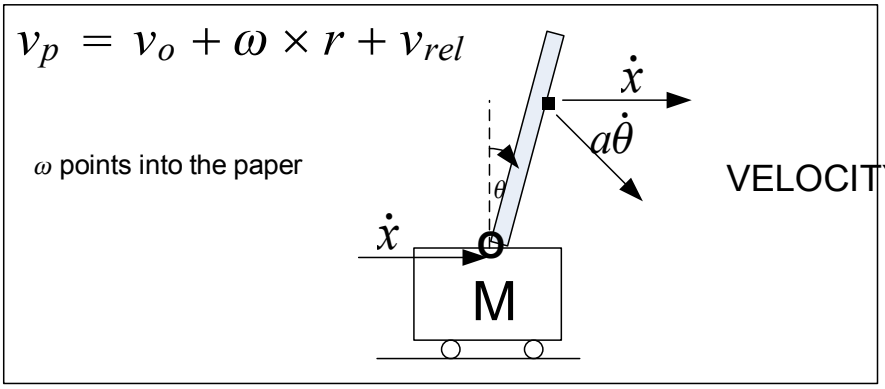
Comparing Eqs. 4,5 to Eqs. 4C,5C, we see they are the same.

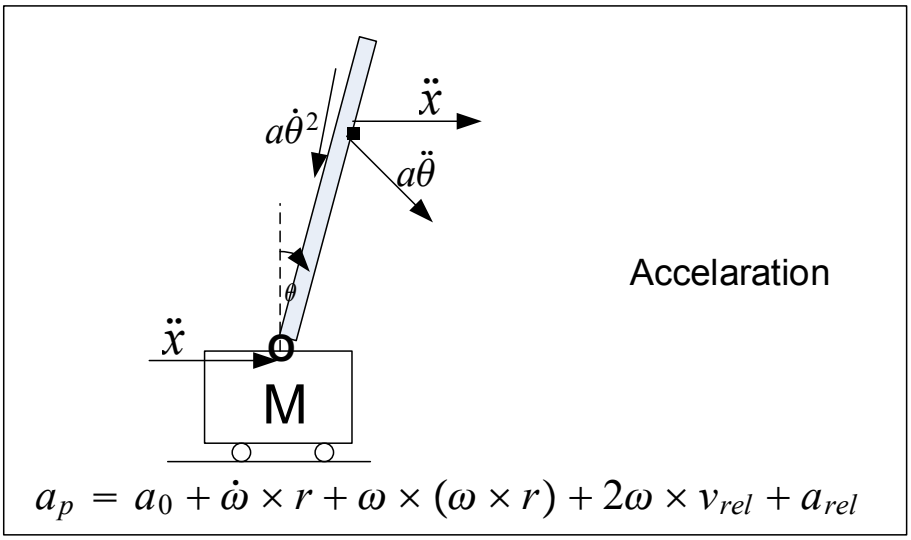
3.1.2 problem 2

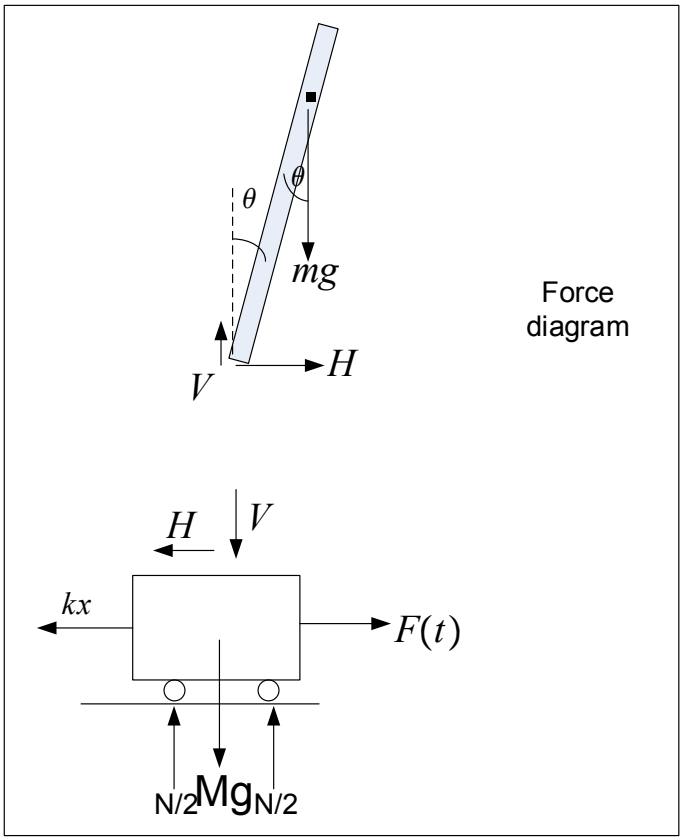
Velocity, acceleration and force diagrams

Generalized coordinates are: x, θ









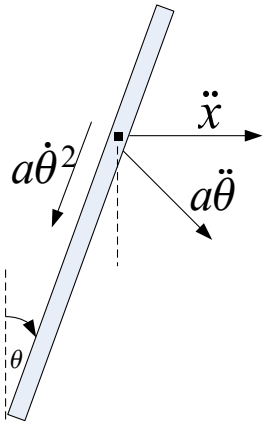
Deriving equations of motion using direct force method

There are 4 unknowns in the system, H, V, x'', θ''. Therefore, we need 4 equations.

Starting with equation of motion for mass M

$$\overrightarrow{\sum} F_x = Mx''$$
$$F(t) - kx - H = Mx'' \tag{1}$$

For rod



$$\overrightarrow{\sum} F_x = m\left(x'' + a\theta'' \cos\theta - a\left(\theta'\right)^2 \sin\theta\right)$$
$$H = m\left(x'' + a\theta'' \cos\theta - a\left(\theta'\right)^2 \sin\theta\right) \tag{2}$$

$$\overrightarrow{\sum} F_y = m\left(-a\theta'' \sin\theta - a\left(\theta'\right)^2 \cos\theta\right)$$
$$V - mg = -m\left(a\theta'' \sin\theta + a\left(\theta'\right)^2 \cos\theta\right) \tag{3}$$

Take moments around c.g., anti-clockwise is positive, hence

$$Ha\cos\theta - Va\sin\theta = -I_g\theta''$$
$$V = \frac{Ha\cos\theta + I_g\theta''}{a\sin\theta}$$

Substituting the above in (3) gives

$$\frac{Ha\cos\theta + I_g\theta''}{a\sin\theta} - mg = -m\left(a\theta'' \sin\theta + a\left(\theta'\right)^2 \cos\theta\right)$$
$$H = \frac{\left[-m\left(a\theta'' \sin\theta + a\left(\theta'\right)^2 \cos\theta\right) + mg\right] a\sin\theta - I_g\theta''}{a\cos\theta}$$

Substituting H from (2) into the above gives

$$m\left(x'' + a\theta'' \cos\theta - a(\theta')^2 \sin\theta\right) = \frac{\left[-m\left(a\theta'' \sin\theta + a\left(\theta'\right)^2 \cos\theta\right) + mg\right] a\sin\theta - I_g\theta''}{a\cos\theta}$$
$$ma\cos\theta x'' + ma^2\cos\theta\theta''\cos\theta - ma^2\cos\theta(\theta')^2\sin\theta = -ma^2\theta''\sin^2\theta - ma^2\sin\theta(\theta')^2\cos\theta + mga\sin\theta - I_g\theta''$$
$$\cos\theta x'' + a\theta''(\cos\theta)^2 - a\cos\theta\sin\theta(\theta')^2 = -a\theta''(\sin\theta)^2 - a\sin\theta\cos\theta(\theta')^2 + g\sin\theta - \frac{I_g\theta''}{ma}$$
$$a\theta'' = g\sin\theta - \cos\theta x'' - \frac{I_g\theta''}{ma}$$
$$\theta''\left(a + \frac{I_g}{ma}\right) = g\sin\theta - \cos\theta x''$$
$$\theta'' = \frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g}$$

Substituting H into Eq. (1) gives

$$F(t) - kx - H = Mx''$$
$$F(t) - kx - m\left(x'' + a\theta'' \cos\theta - a\left(\theta'\right)^2 \sin\theta\right) = Mx''$$
$$x''(M+m) = F(t) - kx - ma\theta''\cos\theta + ma\left(\theta'\right)^2\sin\theta$$
$$x'' = \frac{F(t) - kx - ma\theta''\cos\theta + ma\left(\theta'\right)^2\sin\theta}{M+m}$$

Therefore, the final EQM are

$$x'' = \frac{F(t) - kx - ma\theta''\cos\theta + ma\left(\theta'\right)^2\sin\theta}{M+m} \tag{4}$$
$$\theta'' = \frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g} \tag{5}$$

Decouple the ODE's

Substitute Eq. (5) into Eq. (4) gives

$$x'' = \frac{F(t) - kx - ma\left[\frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g}\right]\cos\theta + ma(\theta')^2\sin\theta}{M+m}$$

Let $\left(ma^2 + I_g\right) = I_o$, hence

$$x'' = \frac{I_oF(t) - I_okx - ma\left[mag\sin\theta - max''\cos\theta\right]\cos\theta + I_oma(\theta')^2\sin\theta}{I_o(M+m)}$$

$$I_o(M+m)x'' - m^2a^2x''\cos\theta = I_oF(t) - I_okx - m^2a^2g\sin\theta\cos\theta + I_oma(\theta')^2\sin\theta$$

$$x'' = \frac{I_oF(t) - I_okx - m^2a^2g\sin\theta\cos\theta + I_oma(\theta')^2\sin\theta}{I_o(M+m) - m^2a^2\cos\theta} \tag{4A}$$

Also, we can solve for θ'' by Substituting Eq. (4) into (5)

$$\theta'' = \frac{mag\sin\theta - ma\left[\frac{F(t) - kx - ma\theta''\cos\theta + ma(\theta')^2\sin\theta}{M+m}\right]\cos\theta}{ma^2 + I_g}$$

Let $\left(ma^2 + I_g\right) = I_o$, hence

$$\theta''I_o(M+m) = (M+m)mag\sin\theta - ma\left[F(t) - kx - ma\theta''\cos\theta + ma(\theta')^2\sin\theta\right]\cos\theta$$

$$= (M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta + m^2a^2\theta''\cos^2\theta - m^2a^2(\theta')^2\sin\theta\cos\theta$$

$$\theta''\left[I_o(M+m) - m^2a^2\cos^2\theta\right] = (M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta$$

$$\theta'' = \frac{(M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta}{I_o(M+m) - m^2a^2\cos^2\theta} \tag{5A}$$

Therefore, the final EQM are

$$x'' = \frac{I_oF(t) - I_okx - m^2a^2g\sin\theta\cos\theta + I_oma(\theta')^2\sin\theta}{I_o(M+m) - m^2a^2\cos\theta}$$

$$\theta'' = \frac{(M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta}{I_o(M+m) - m^2a^2\cos^2\theta}$$

Convert to state space form

Using the decoupled ODE's above, Eqs (4A) and (5A), and introducing 4 state variables $x_1, x_2, x_3 x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) - I_o k x - m^2 a^2 g \sin\theta \cos\theta + I_o m a (\theta')^2 \sin\theta}{I_o(M+m) - m^2 a^2 \cos\theta} \\ \frac{(M+m) mag \sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2 a^2 (\theta')^2 \sin\theta \cos\theta}{I_o(M+m) - m^2 a^2 \cos^2\theta} \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) - I_o k x_1 - m^2 a^2 g \sin x_2 \cos x_2 + I_o m a\ x_4^2 \sin x_2}{I_o(M+m) - m^2 a^2 \cos x_2} \\ \frac{(M+m) mag \sin x_2 - maF(t)\cos x_2 + makx\cos x_2 - m^2 a^2 x_4^2 \sin x_2 \cos x_2}{I_o(M+m) - m^2 a^2 \cos^2 x_2} \end{pmatrix}$$

Deriving equations of motion using Lagrangian method (for verification)

To verify the solution above, try using Lagrangian method. Let L be the Lagrangian, and let T be the kinetic energy of the system, and U the potential energy.

$$L = T - U$$

$$T = \frac{1}{2}M\dot{x}^2 + \overbrace{\frac{1}{2}I_g\dot{\theta}^2}^{\text{angular bar K.E.}} + \overbrace{\frac{1}{2}m\left(\dot{x}^2 + \left(a\dot{\theta}\right)^2 + 2\dot{x}a\dot{\theta}\cos\theta\right)}^{\text{linear bar K.E. of its c.g.}}$$

For the potential energy, the mass M has

$$U_M = \frac{1}{2}kx^2$$

and the bar is losing PE, hence, assuming that the zero potential energy is at the ground level

$$U_m = mga\cos\theta$$

Therefore, L becomes

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}I_g\dot{\theta}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(a\dot{\theta}\right)^2 + 2\dot{x}a\dot{\theta}\cos\theta\right) - \left(\frac{1}{2}kx^2 + mga\cos\theta\right)$$

To obtain equations of motion, evaluate

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F \tag{1A}$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{2A}$$

For M we obtain

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} + ma\dot{\theta}\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = M\ddot{x} + m\ddot{x} + ma\ddot{\theta}\cos\theta - ma\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x} = -kx$$

Hence, for the mass M, the equation of motion is

$$M\ddot{x} + m\ddot{x} + ma\ddot{\theta}\cos\theta - ma\dot{\theta}^2\sin\theta + kx = F$$

$$\ddot{x} = \frac{F - kx - ma\ddot{\theta}\cos\theta + ma\dot{\theta}^2\sin\theta}{M + m} \tag{4C}$$

To find EQM for mass m, evaluate $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$

$$\frac{\partial L}{\partial \dot{\theta}} = I_g\dot{\theta} + m\left(a^2\dot{\theta} + \dot{x}a\cos\theta\right)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = I_g\ddot{\theta} + m\left(a^2\ddot{\theta} - \dot{x}a\dot{\theta}\sin\theta + \ddot{x}a\cos\theta\right)$$

$$\frac{\partial L}{\partial \theta} = -m\dot{x}a\dot{\theta}\sin\theta + mga\sin\theta$$

Hence EQM is

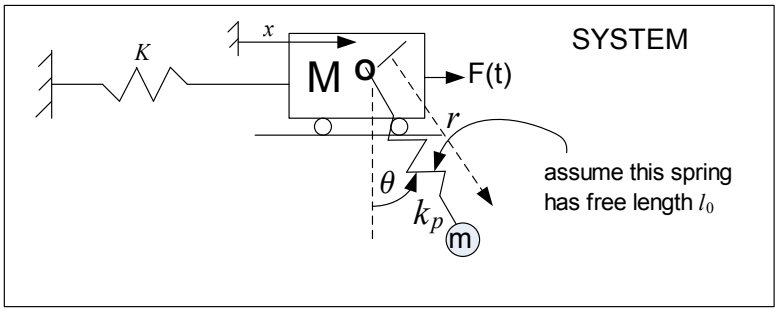
$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} &= 0 \\
I_g\ddot{\theta} + m\left(a^2\ddot{\theta} - \dot{x}a\dot{\theta}\sin\theta + \ddot{x}a\cos\theta\right) + m\dot{x}a\dot{\theta}\sin\theta - mga\sin\theta &= 0 \\
I_g\ddot{\theta} + ma^2\ddot{\theta} - m\dot{x}a\dot{\theta}\sin\theta + m\ddot{x}a\cos\theta + m\dot{x}a\dot{\theta}\sin\theta - mga\sin\theta &= 0 \\
I_g\ddot{\theta} + ma^2\ddot{\theta} + m\ddot{x}a\cos\theta - mga\sin\theta &= 0 \\
\ddot{\theta}\left(I_g + ma^2\right) &= mga\sin\theta - m\ddot{x}a\cos\theta \\
\ddot{\theta} &= \frac{mga\sin\theta - m\ddot{x}a\cos\theta}{I_g + ma^2} \\
&= \frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g}
\end{aligned} \tag{5C}$$

Compare (4A,5A) to (4C,5C) we see they are the same.

3.1.3 Solution problem 3

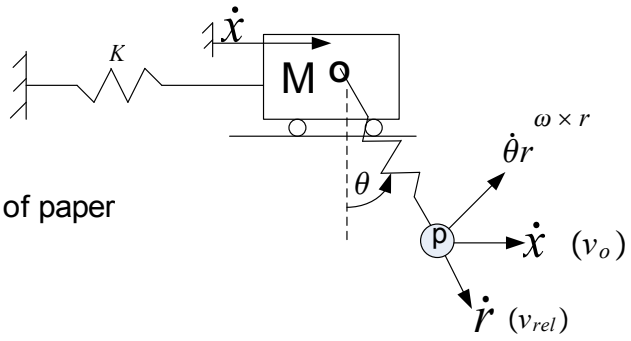
Velocity, acceleration and force diagrams

Generalized coordinates are: x, θ, r

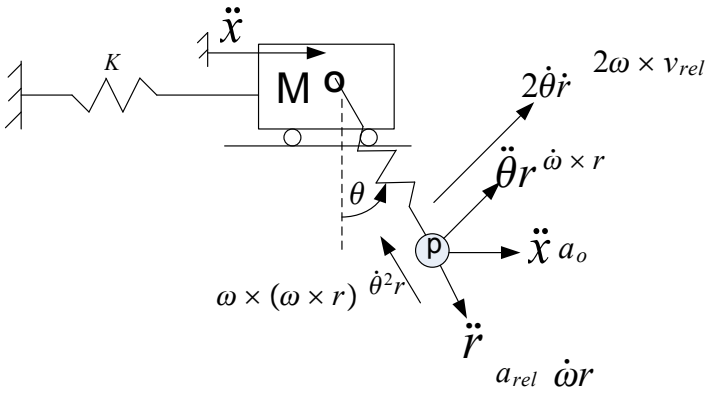


$$v_p = v_o + \omega \times r + v_{rel}$$

Velocity

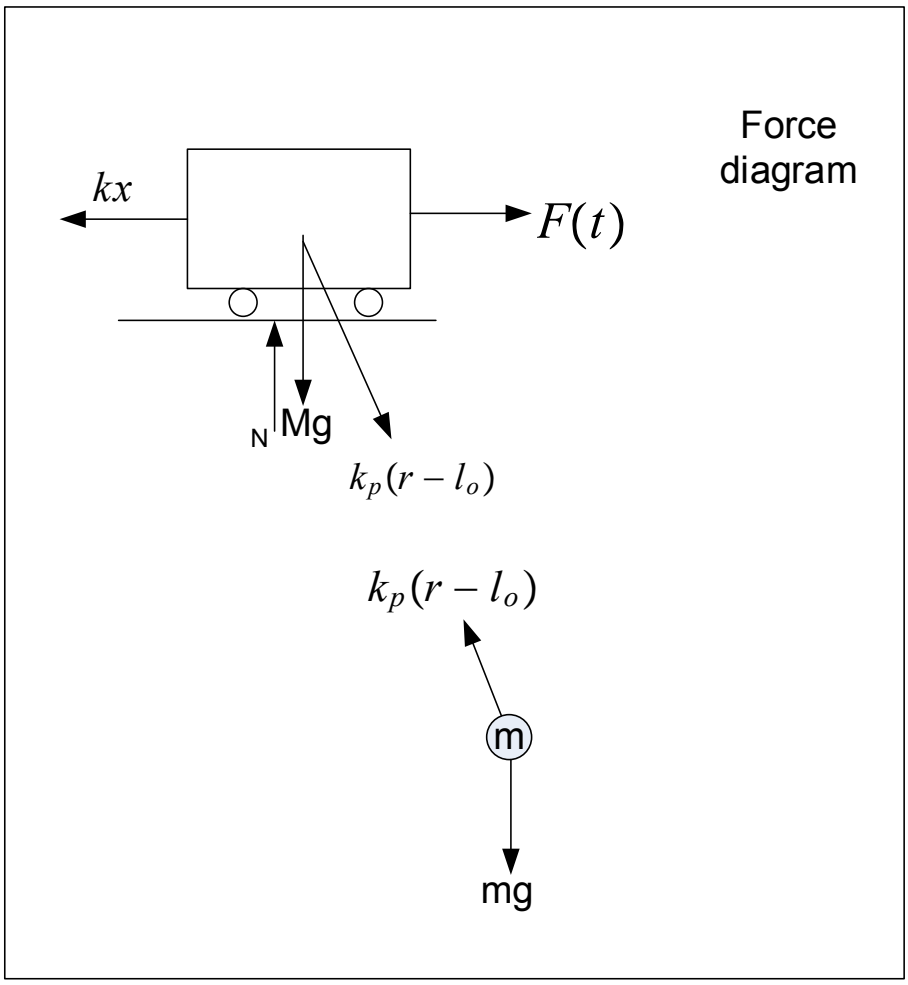


ω points out of paper



Accelaration

$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$



Deriving equations of motion using direct force method

There are 4 unknowns in the system, x'', θ'', r''. Therefore, we need 3 equations.

Equation of motion for M

$$\overrightarrow{\sum} F_x = Mx''$$
$$F(t) - kx - k_p(r - l_0)\sin\theta = Mx'' \tag{1}$$

Equation of motion for m, radial direction

$$\searrow \sum F = m\left(r'' + x''\sin\theta - \dot{\theta}^2 r\right)$$
$$mg\cos\theta - k_p(r - l_0) = m\left(r'' + x''\sin\theta - \dot{\theta}^2 r\right) \tag{2}$$

Equation of motion for m, θ direction

$$\circlearrowleft \sum F = m\left(\theta'' r + 2r'\theta' + x'' \cos\theta\right)$$
$$-mg\sin\theta = m\left(\theta'' r + 2r'\theta' + x'' \cos\theta\right) \tag{3}$$

The above are the 3 EQM needed. Now we simplify them. From Eq. (1)

$$x'' = \frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta \tag{1A}$$

From Eq. (2)

$$r'' = g\cos\theta - \frac{k_p}{m}(r - l_0) - x''\sin\theta + \dot{\theta}^2 r \tag{2A}$$

and from Eq. (3)

$$\theta'' = -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{x''}{r}\cos\theta \tag{3A}$$

Decouple the ODE's

Substitute Eq. (1A) into (2A)

$$\begin{aligned} r'' &= g\cos\theta - \frac{k_p}{m}(r - l_0) - \left[\frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta\right]\sin\theta + \dot{\theta}^2 r \\ &= g\cos\theta - \frac{k_p}{m}(r - l_0) - \frac{F(t)}{M}\sin\theta + \frac{k}{M}x\sin\theta + \frac{k_p}{M}(r - l_0)\sin^2\theta + \dot{\theta}^2 r \end{aligned}$$

And Substitute Eq. (1A) into (3A)

$$\begin{aligned} \theta'' &= -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{1}{r}\left[\frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta\right]\cos\theta \\ &= -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{F(t)}{rM}\cos\theta + \frac{k}{rM}x\cos\theta + \frac{k_p}{rM}(r - l_0)\sin\theta\cos\theta \end{aligned}$$

Therefore, the final EQM are

$$\begin{aligned} x'' &= \frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta \\ r'' &= g\cos\theta - \frac{k_p}{m}(r - l_0) - \frac{F(t)}{M}\sin\theta + \frac{k}{M}x\sin\theta + \frac{k_p}{M}(r - l_0)\sin^2\theta + \dot{\theta}^2 r \\ \theta'' &= -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{F(t)}{rM}\cos\theta + \frac{k}{rM}x\cos\theta + \frac{k_p}{rM}(r - l_0)\sin\theta\cos\theta \end{aligned}$$

Convert to state space form

Using the decoupled ODE's above, and introducing 6 state variables $x_1, x_2, x_{3,} x_4, x_5, x_6$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ r \\ x' \\ \theta' \\ r' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ r' \\ x'' \\ \theta'' \\ r'' \end{pmatrix}$$

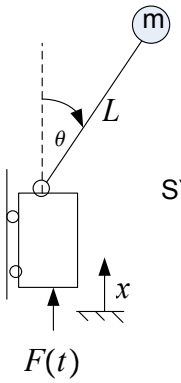
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta \\ -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{F(t)}{rM}\cos\theta + \frac{k}{rM}x\cos\theta + \frac{k_p}{rM}(r - l_0)\sin\theta\cos\theta \\ g\cos\theta - \frac{k_p}{m}(r - l_0) - \frac{F(t)}{M}\sin\theta + \frac{k}{M}x\sin\theta + \frac{k_p}{M}(r - l_0)\sin^2\theta + \dot{\theta}^2 r \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{F(t)}{M} - \frac{k}{M}x_1 - \frac{k_p}{M}(x_3 - l_0)\sin x_2 \\ -\frac{g}{r}\sin x_2 - \frac{2x_6}{x_3}x_5 - \frac{F(t)}{x_3 M}\cos x_2 + \frac{k}{x_3 M}x_1\cos x_2 + \frac{k_p}{x_3 M}(x_3 - l_0)\sin x_2\cos x_2 \\ g\cos x_2 - \frac{k_p}{m}(x_3 - l_0) - \frac{F(t)}{M}\sin x_2 + \frac{k}{M}x_1\sin x_2 + \frac{k_p}{M}(x_3 - l_0)\sin^2 x_2 + x_5^2 x_3 \end{pmatrix}$$

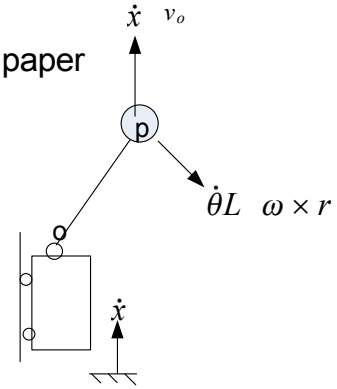
3.1.4 Solution problem 4

Velocity, acceleration and force diagrams

Generalized coordinates are: x, θ



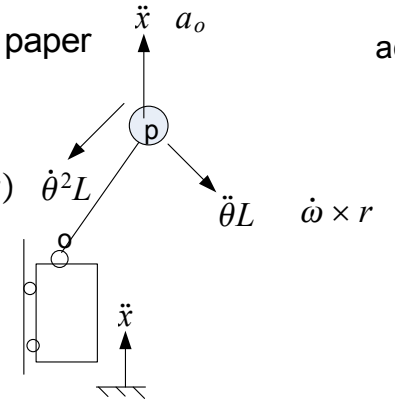
SYSTE
M



ω is into the paper

Velocity

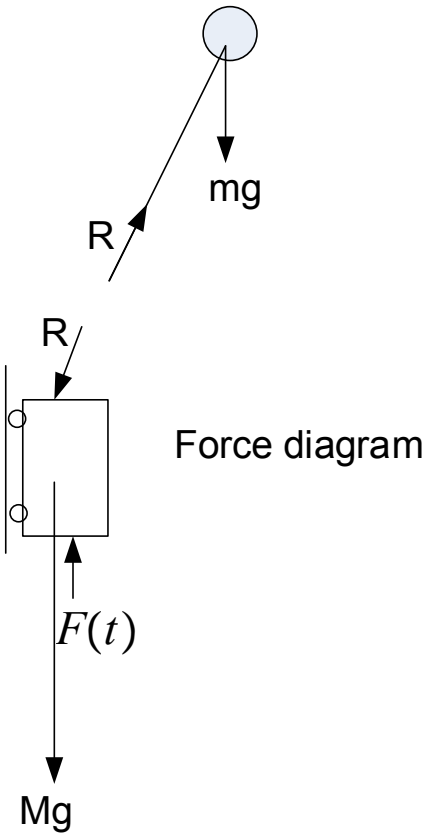
$$v_p = v_o + \omega \times r + v_{rel}$$



ω is into the paper

acceleration

$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$



Deriving equations of motion using direct force method

There are 3 unknowns in the system, x'', θ'', R Therefore, we need 3 equations.

Equation of motion for M

$$\uparrow \sum F = Mx''$$
$$F(t) - Mg - R\cos\theta = Mx'' \tag{1}$$

Equation of motion for m, along θ

$$\circlearrowleft \sum F = m\left(-x''\sin\theta + \theta'' L\right)$$
$$mg\sin\theta = m\left(-x''\sin\theta + \theta'' L\right)$$

Equation of motion for m, along r

$$\nearrow \sum F = 0$$
$$R - mg\cos\theta = m\left(x''\cos\theta - (\theta')^2 L\right) \tag{3}$$

From Eq. (2) we find EQM for θ''

$$\theta'' = \frac{g}{L}\sin\theta + \frac{x''}{L}\sin\theta \tag{4}$$

From Eq. (3) find R and substitute result into (1).

$$R = mg\cos\theta + m\left(x''\cos\theta - (\theta')^2 L\right)$$

Hence Eq. (1) becomes

$$F(t) - Mg - \left[mg\cos\theta + m\left(x''\cos\theta - (\theta')^2 L\right)\right]\cos\theta = Mx''$$

Therefore

$$\begin{aligned}
x'' &= \frac{F(t)}{M} - g - \frac{m}{M}g\cos^2\theta - \frac{m}{M}\left(x''\cos^2\theta - \cos\theta\,(\theta')^2 L\right) \\
x'' + \frac{m}{M}x''\cos^2\theta &= \frac{F(t)}{M} - g - \frac{m}{M}g\cos^2\theta + \frac{m}{M}\cos\theta\,(\theta')^2 L \\
x''\left(1 + \frac{m}{M}\cos^2\theta\right) &= \frac{F(t)}{M} - g - \frac{m}{M}g\cos^2\theta + \frac{m}{M}\cos\theta\,(\theta')^2 L \\
x''\left(M + m\cos^2\theta\right) &= F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L \\
x'' &= \frac{F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L}{M + m\cos^2\theta} \qquad (5)
\end{aligned}$$

Therefore, the EQM are

$$\begin{aligned}
x'' &= \frac{F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L}{M + m\cos^2\theta} \\
\theta'' &= \frac{g}{L}\sin\theta + \frac{x''}{L}\sin\theta
\end{aligned}$$

Decouple the ODE's

$$x'' = \frac{F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L}{M + m\cos^2\theta}$$

and

$$\begin{aligned}
\theta'' &= \frac{g}{L}\sin\theta + \frac{x''}{L}\sin\theta \\
&= \frac{g}{L}\sin\theta + \frac{\sin\theta}{L}\left[\frac{F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L}{M + m\cos^2\theta}\right] \\
&= \frac{g}{L}\sin\theta + \frac{F(t)\sin\theta - gM\sin\theta - mg\cos^2\theta\sin\theta + m\cos\theta\,(\theta')^2 L\sin\theta}{L\left(M + m\cos^2\theta\right)}
\end{aligned}$$

Therefore, the final EQM are

$$\begin{aligned}
x'' &= \frac{F(t) - gM - mg\cos^2\theta + m\cos\theta\,(\theta')^2 L}{M + m\cos^2\theta} \\
\theta'' &= \frac{g}{L}\sin\theta + \frac{F(t)\sin\theta - gM\sin\theta - mg\cos^2\theta\sin\theta + m\cos\theta\,(\theta')^2 L\sin\theta}{L\left(M + m\cos^2\theta\right)}
\end{aligned}$$

Convert to state space form

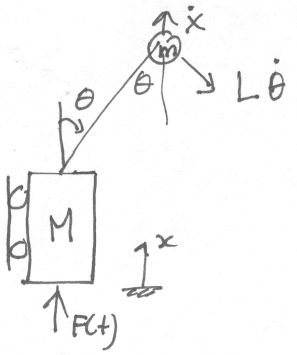
Using the decoupled ODE's above, and introducing 4 state variables $x_1, x_2, x_{3,} x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t) - gM - mg\cos^2\theta + m\cos\theta(\theta')^2 L}{M + m\cos^2\theta} \\ \frac{g}{L}\sin\theta + \frac{F(t)\sin\theta - gM\sin\theta - mg\cos^2\theta\sin\theta + m\cos\theta(\theta')^2 L\sin\theta}{L(M + m\cos^2\theta)} \end{pmatrix}$$

$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t) - gM - mg\cos^2 x_2 + m\cos x_2 x_4^2 L}{M + m\cos^2 x_2} \\ \frac{g}{L}\sin x_2 + \frac{F(t)\sin x_2 - gM\sin x_2 - mg\cos^2 x_2\sin x_2 + m\cos x_2 x_4^2 L\sin x_2}{L(M + m\cos^2 x_2)} \end{pmatrix}$$

Solving using Lagrange



$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left((\dot{x}-L\dot{\theta})^2 + (L\dot{\theta}\cos\theta)^2\right)$$

$$V = Mgx + mg(L\cos\theta + x)$$

$$L = T - V = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + L^2\dot{\theta}^2 - 2\dot{x}\dot{\theta}L + L^2\dot{\theta}^2\cos^2\theta\right) - Mgx - mgL\cos\theta - mgx$$

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} - m\dot{\theta}L$$

$$\frac{d}{dt}(\cdot) = M\ddot{x} + m\ddot{x} - m\ddot{\theta}L$$

$$\frac{\partial L}{\partial x} = -Mg - mg$$

$$\Rightarrow (m+M)\ddot{x} - m\ddot{\theta}L + g(M+m) = F_x$$

$$\delta W = \frac{F(t)\,\delta x}{\delta x} \Rightarrow F_x = F(t)$$

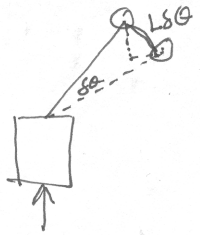
$$\text{Sv}\left[\ddot{x} = \frac{F(t) - g(M+m) + m\ddot{\theta}L}{(m+M)}\right]$$

for θ $\quad \frac{\partial L}{\partial \dot\theta} = mL^2\dot\theta - m\dot x L + L^2\dot\theta\cos^2\theta$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot\theta}\right) = mL^2\ddot\theta - m\ddot x L + L^2\ddot\theta\cos^2\theta - 2L^2\dot\theta\cos\theta\sin\theta\,\dot\theta$$
$$= mL^2\ddot\theta - m\ddot x L + L^2\ddot\theta\cos^2\theta - 2L^2\dot\theta^2\cos\theta\sin\theta$$

$$\frac{\partial L}{\partial \theta} = +mL^2\dot\theta^2\sin\theta + mgL\sin\theta$$

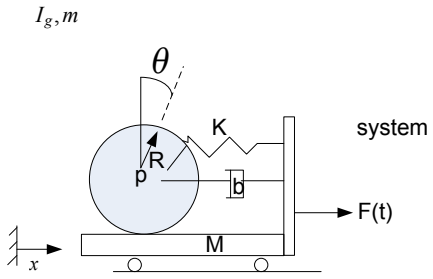
so $\quad mL^2\ddot\theta - m\ddot x L + L^2\ddot\theta\cos^2\theta - 2L^2\dot\theta^2\cos\theta\sin\theta$
$$-mL^2\dot\theta^2\sin\theta - mgL\sin\theta = F_\theta.$$

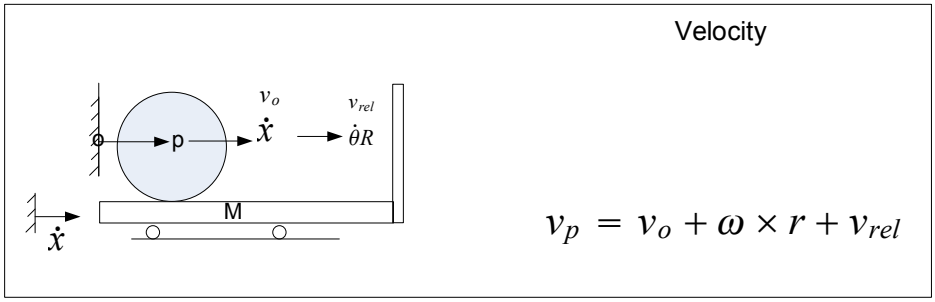


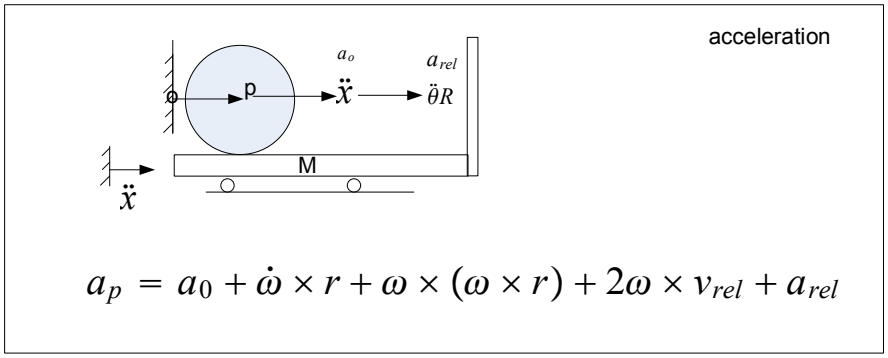
no direct external force on m. so work $= 0$. F_θ.

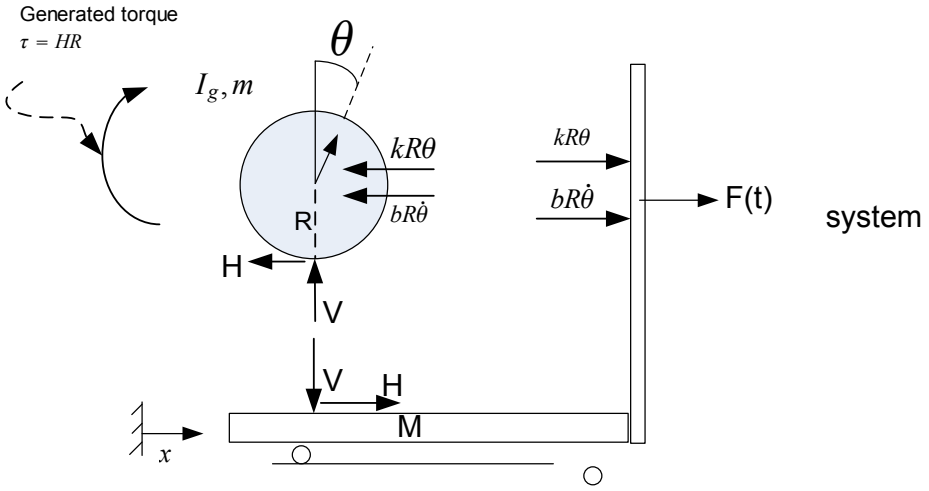
$$\therefore \quad \ddot\theta = \frac{mL^2\dot\theta^2\sin\theta + mgL\sin\theta + 2L^2\dot\theta^2\cos\theta\sin\theta + m\ddot x L}{(mL^2 + L^2\cos^2\theta)}$$

3.1.5 Solution problem 5









The vertical reaction V is not needed since there is no acceleration in the vertical direction. The above diagram shows forces assuming the spring is in compression. The unknowns are θ'', x'', H, hence we need 3 equations.

For mass M

$$\rightarrow \sum F = Mx''$$
$$F(t) + kR\theta + bR\theta' + H = Mx''$$

or mass m

$$\rightarrow \sum F = 0$$

$$-kR\theta - bR\theta' - H = m(x'' + \theta''R)$$

and moments around the origin of the disk, anti-clockwise positive

$$-HR = -I_g\theta''$$
$$H = \frac{I_g\theta''}{R} \tag{3}$$

Substitute (3) into (2) gives

$$-kR^2\theta - bR^2\theta' - I_g\theta'' = mR(x'' + \theta''R)$$
$$\theta''\left(I_g + mR^2\right) = -mRx'' - kR^2\theta - bR^2\theta'$$
$$\theta'' = \frac{-\left(mRx'' + kR^2\theta + bR^2\theta'\right)}{I_g + mR^2}$$

Let $I_g + mR^2 = I_o$ hence the above becomes

$$\theta'' = -\frac{mRx'' + kR^2\theta + bR^2\theta'}{I_o}$$

Therefore, the EQM are

$$Mx'' = F(t) + kR\theta + bR\theta' + H$$
$$x'' = \frac{F(t) + kR\theta + bR\theta' + \frac{I_g\theta''}{R}}{M}$$
$$x'' = \frac{RF(t) + kR^2\theta + bR^2\theta' - I_g\theta''}{MR} \tag{4}$$
$$\theta'' = -\frac{mRx'' + kR^2\theta + bR^2\theta'}{I_o} \tag{5}$$

Decouple the ODE's

Substitute Eq. (5) into (4) gives

$$x'' = \frac{RF(t) + kR^2\theta + bR^2\theta' - I_g\left[-\frac{mRx'' + kR^2\theta + bR^2\theta'}{I_o}\right]}{MR}$$
$$x''MRI_o = I_oRF(t) + kI_oR^2\theta + bI_oR^2\theta' + I_g\left[mRx'' + kR^2\theta + bR^2\theta'\right]$$
$$x'' = \frac{I_oF(t) + kI_oR\theta + bI_oR\theta' + I_g\left[kR\theta + bR\theta'\right]}{MI_o - I_gm}$$

Now to decouple the θ'', substituting Eq. (4) into (5) gives

$$\theta'' = -\frac{mR\left[\frac{RF(t)+kR^2\theta+bR^2\theta'-I_g\theta''}{MR}\right] + kR^2\theta + bR^2\theta'}{I_o}$$

$$\theta'' I_o = -mR\left[\frac{RF(t) + kR^2\theta + bR^2\theta' - I_g\theta''}{MR}\right] - kR^2\theta - bR^2\theta'$$

$$\theta''\left(I_oM - mI_g\right) = -mRF(t) - mkR^2\theta - mbR^2\theta' - MkR^2\theta - MbR^2\theta'$$

$$\theta'' = \frac{-mRF(t) - mkR^2\theta - mbR^2\theta' - MkR^2\theta - MbR^2\theta'}{MI_o - I_gm}$$

Therefore, the final EQM are

$$x'' = \frac{I_oF(t) + kI_oR\theta + bI_oR\theta' + I_g\left[kR\theta + bR\theta'\right]}{MI_o - I_gm}$$

$$\theta'' = \frac{-mRF(t) - mkR^2\theta - mbR^2\theta' - MkR^2\theta - MbR^2\theta'}{MI_o - I_gm}$$

Convert to state space form

Using the decoupled ODE's above, and introducing 4 state variables x_1, x_2, x_3, x_4 we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_oF(t)+kI_oR\theta+bI_oR\theta'+I_g[kR\theta+bR\theta']}{MI_o-I_gm} \\ \frac{-mRF(t)-mkR^2\theta-mbR^2\theta'-MkR^2\theta-MbR^2\theta'}{MI_o-I_gm} \end{pmatrix}$$

$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_oF(t)+kI_oRx_2+bI_oRx_4+I_g[kRx_2+bRx_4]}{MI_o-I_gm} \\ \frac{-mRF(t)-mkR^2x_2-mbR^2x_4-MkR^2x_2-MbR^2x_4}{MI_o-I_gm} \end{pmatrix}$$

Solving using Lagrangian method

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x} + R\dot{\theta}\right)^2 + \frac{1}{2}I_g\dot{\theta}^2$$

$$V = \frac{1}{2}k\left(R\theta\right)^2$$

Hence

$$\begin{aligned} L &= T - V \\ &= \left[\frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x} + R\dot{\theta}\right)^2 + \frac{1}{2}I_g\dot{\theta}^2\right] - \frac{1}{2}k\left(R\theta\right)^2 \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + R^2\dot{\theta}^2 + 2\dot{x}\dot{\theta}R\right) + \frac{1}{2}I_g\dot{\theta}^2 - \frac{1}{2}kR^2\theta^2 \end{aligned}$$

and

$$\begin{aligned} \frac{\partial L}{\partial \dot{x}} &= M\dot{x} + m\dot{x} + m\dot{\theta}R \\ \frac{\partial L}{\partial x} &= 0 \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= M\ddot{x} + m\ddot{x} + m\ddot{\theta}R \end{aligned}$$

Hence, for x, the EQM is

$$M\ddot{x} + m\ddot{x} + m\ddot{\theta}R = Q_x$$

Where Q_x is the genralized force which is

$$\begin{aligned} \delta W &= \frac{F\delta x + \left(kR\theta\right)\delta x + \left(bR\dot{\theta}\right)\delta x}{\delta x} \\ &= F + bR\dot{\theta} + kR\theta \end{aligned}$$

hence

$$\begin{aligned} M\ddot{x} + m\ddot{x} + m\ddot{\theta}R &= F\left(t\right) + bR\dot{\theta} + kR\theta \\ \ddot{x} &= \frac{F\left(t\right) + bR\dot{\theta} + kR\theta - m\ddot{\theta}R}{M + m} \end{aligned}$$

Since $I_g = \frac{mR^2}{2}$, then the above can be written as

$$\ddot{x} = \frac{RF\left(t\right) + bR^2\dot{\theta} + kR^2\theta - 2I_g\ddot{\theta}}{R\left(M + m\right)}$$

For θ, the EQM is

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}} &= m\left(R^2\dot{\theta} + \dot{x}R\right) + I_g\dot{\theta} \\ \frac{\partial L}{\partial \theta} &= -kR^2\theta \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} \end{aligned}$$

Hence

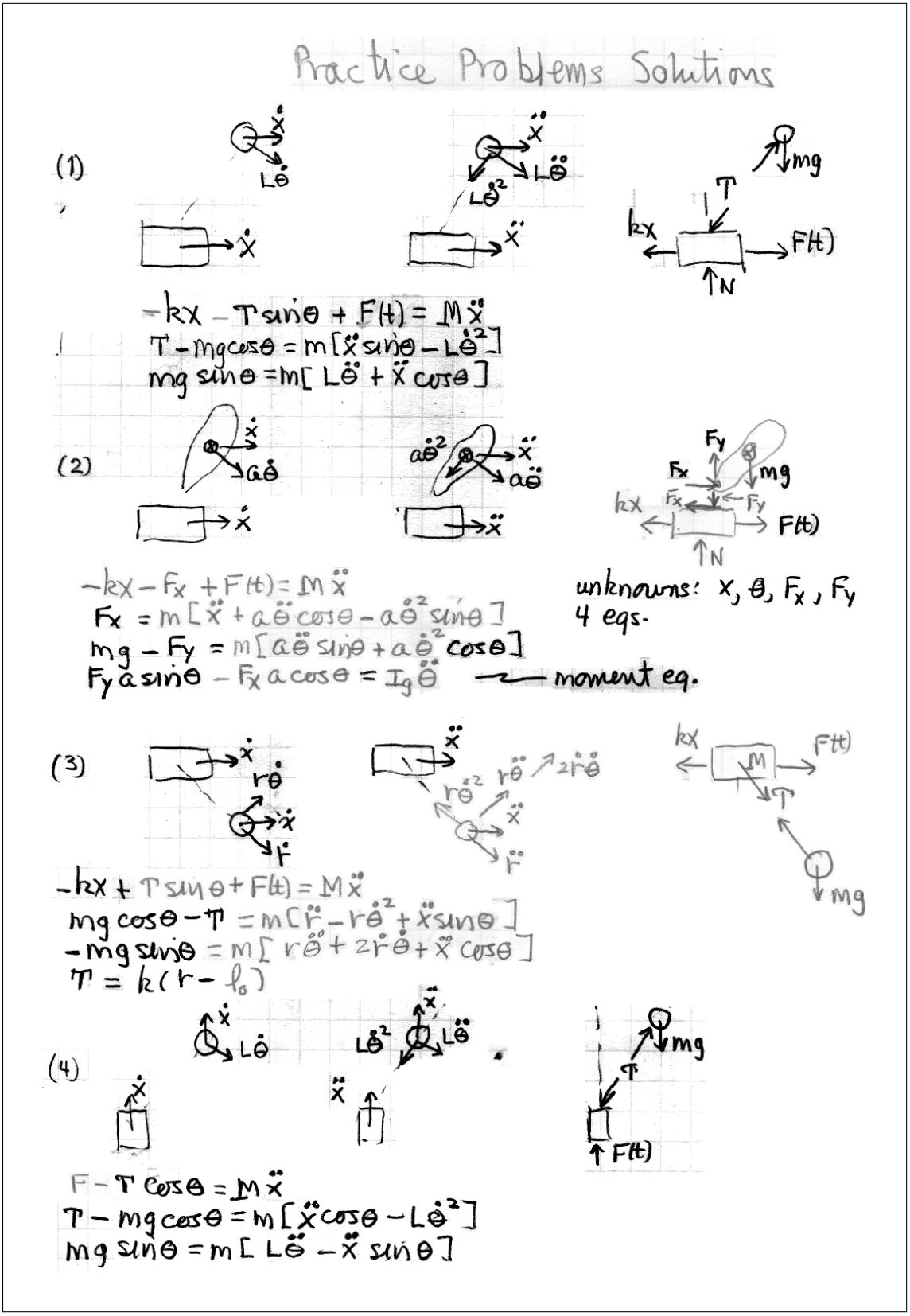
$$m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} + kR^2\theta = F_\theta$$

In this case, $\delta W = -\frac{\left(bR\dot{\theta}\right)R\delta\theta}{\delta\theta}$, hence $F_\theta = -bR^2\dot{\theta}$, therefore, the EQM is

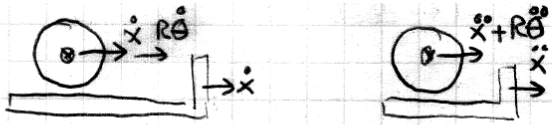
$$\begin{aligned} m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} + kR^2\theta &= -bR^2\dot{\theta} \\ \ddot{\theta} &= \frac{-bR^2\dot{\theta} - kR^2\theta - m\ddot{x}R}{I_g + mR^2} \\ &= -\frac{bR^2\dot{\theta} + kR^2\theta + m\ddot{x}R}{I_o} \end{aligned}$$

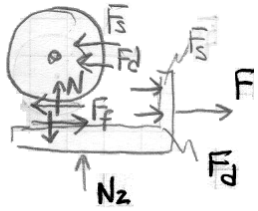
Which is the same as Eq. (5) above.

3.1.6 Key solution



(5)





$$F + F_f + F_s + F_d = M\ddot{x}$$

$$-F_f - F_s - F_d = m[\ddot{x} + R\ddot{\theta}]$$

$$F_f R = I_g \ddot{\theta}$$

$$F_s = k(R\theta)$$

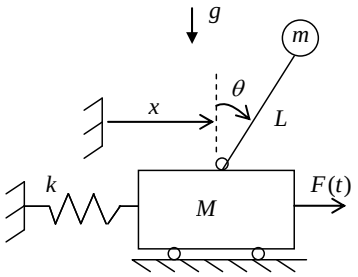
$$F_d = b(R\dot{\theta})$$

3.2 Second HW set, Solve the first set using Lagrangian

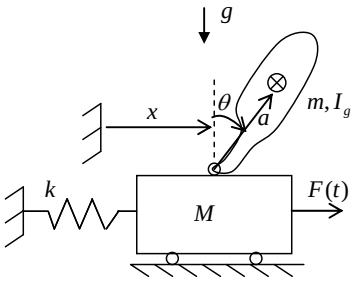
solve the first practice problems using Lagrangian approach.

EME 121 Practice problems

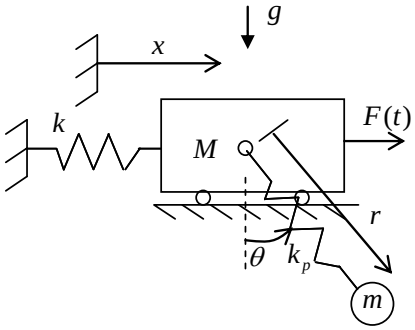
Make velocity, acceleration and force diagrams for these problems and derive the equations of motion using Newton's laws. Present results with highest order derivatives on the left and the lower order terms on the right. Reduce to first order form.

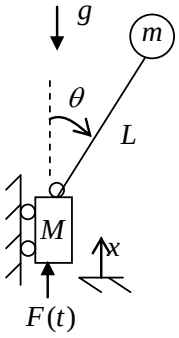


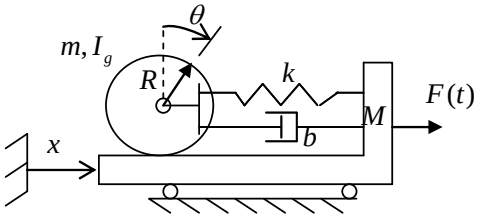
When x=0, spring is relaxed



When x=0, spring is relaxed







Cylinder rolls without slip on cart

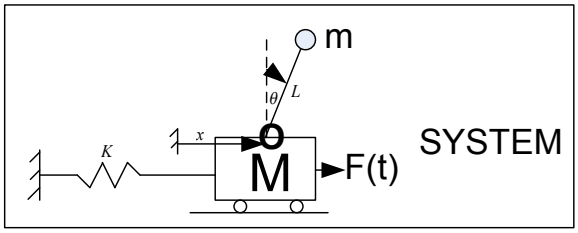
Spring/damper attached to center of cylinder

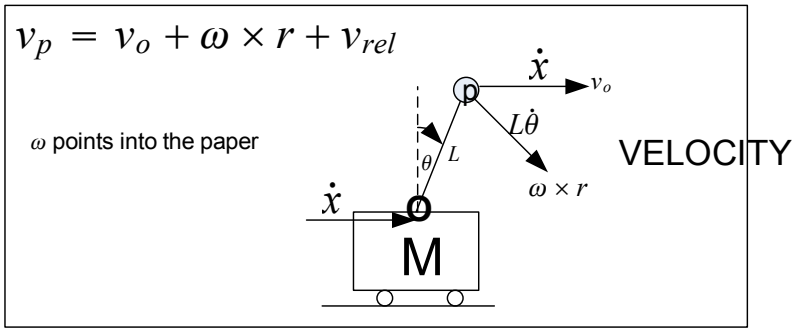
When θ is zero, spring is relaxed

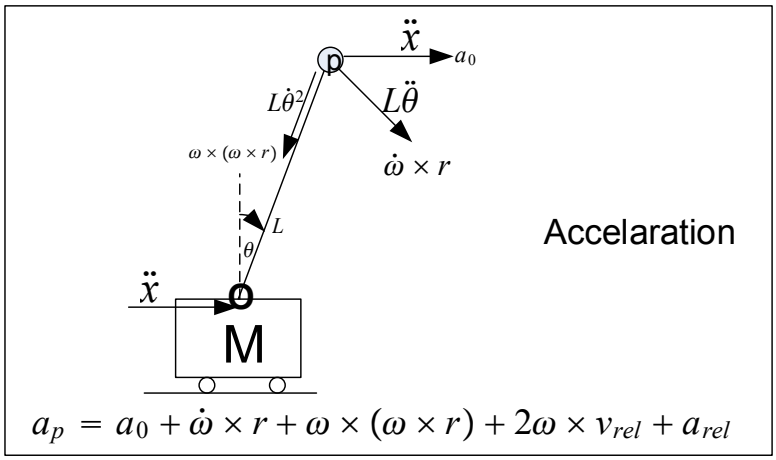
3.2.1 problem 1

Velocity diagram

Only the velocity diagram is needed for the Lagrangian method. The generalized coordinates are: x, θ





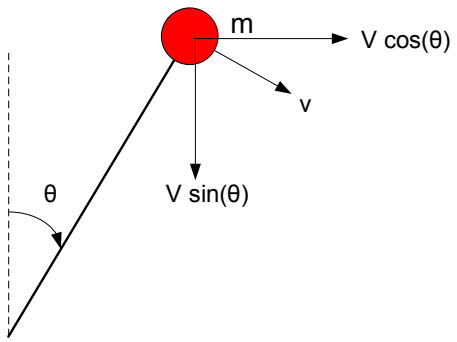


Let L be the Lagrangian, and let T be the kinetic energy of the system, and V the potential energy.

$$L = T - V$$

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}mv^2$$

Where v is the speed of the mass m which is



Hence

$$\begin{aligned} T &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\left(\dot{x} + L\dot{\theta}\cos\theta\right)^2 + \left(L\dot{\theta}\sin\theta\right)^2\right) \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(L\dot{\theta}\right)^2 + 2\dot{x}L\dot{\theta}\cos\theta\right) \end{aligned}$$

For the potential energy

$$V = \frac{1}{2}kx^2 + mgL\cos\theta$$

Therefore, L becomes

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(L\dot{\theta}\right)^2 + 2\dot{x}L\dot{\theta}\cos\theta\right) - \left(\frac{1}{2}kx^2 + mgL\cos\theta\right) \end{aligned}$$

To obtain equations of motion, we evaluate

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = Q_x$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = Q_\theta$$

For the generalized forces, we can readily see that $Q_x = F$ and $Q_\theta = 0$. Hence for x we write

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} + mL\dot{\theta}\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = M\ddot{x} + m\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x} = -kx$$

Hence, for the mass M, the equation of motion is

$$\begin{aligned} M\ddot{x} + m\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta + kx &= F \\ \ddot{x} = \frac{F - mL\ddot{\theta}\cos\theta + mL\dot{\theta}^2\sin\theta - kx}{M + m} \end{aligned} \tag{1}$$

To find EQM for mass m, evaluate $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$

$$\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} + m\dot{x}L\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = mL^2\ddot{\theta} + m\ddot{x}L\cos\theta - m\dot{x}L\dot{\theta}\sin\theta$$

$$\frac{\partial L}{\partial \theta} = -m\dot{x}L\dot{\theta}\sin\theta + mgL\sin\theta$$

Hence EQM is

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} &= 0 \\
mL^2\ddot{\theta} + m\ddot{x}L\cos\theta - m\dot{x}L\dot{\theta}\sin\theta - \left(-m\dot{x}L\dot{\theta}\sin\theta + mgL\sin\theta\right) &= 0 \\
L\ddot{\theta} + \ddot{x}\cos\theta - \dot{x}\dot{\theta}\sin\theta + \dot{x}\dot{\theta}\sin\theta - g\sin\theta &= 0 \\
L\ddot{\theta} + \ddot{x}\cos\theta - g\sin\theta &= 0 \\
\ddot{\theta} = \frac{g\sin\theta - \ddot{x}\cos\theta}{L} \qquad (2)
\end{aligned}$$

3.2.2 Decouple the ODE's

Substitute Eq. (2) into Eq. (1) gives

$$\begin{aligned}
x'' &= \frac{F(t) - kx - mL\left[\frac{g\sin\theta - x''\cos\theta}{L}\right]\cos\theta + mL(\theta')^2\sin\theta}{M+m} \\
&= \frac{F(t) - kx - mg\sin\theta\cos\theta + mx''\cos^2\theta + mL(\theta')^2\sin\theta}{M+m} \\
x''(M+m) - mx''\cos^2\theta &= F(t) - kx - mg\sin\theta\cos\theta + mL(\theta')^2\sin\theta \\
x'' &= \frac{F(t) - kx - mg\sin\theta\cos\theta + mL(\theta')^2\sin\theta}{(M+m-m\cos^2\theta)} \qquad (3)
\end{aligned}$$

Also, we can solve for θ'' by Substituting Eq. (1) into (2)

$$\begin{aligned}
\theta'' &= \frac{g\sin\theta - \left(\frac{F(t) - kx - mL\theta''\cos\theta + mL(\theta')^2\sin\theta}{M+m}\right)\cos\theta}{L} \\
L\theta''(M+m) &= g\sin\theta(M+m) - \left(F(t) - kx - mL\theta''\cos\theta + mL(\theta')^2\sin\theta\right)\cos\theta \\
L\theta''(M+m) &= g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta + mL\theta''\cos^2\theta - mL(\theta')^2\sin\theta\cos\theta \\
L\theta''(M+m) - mL\theta''\cos^2\theta &= g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta - mL(\theta')^2\sin\theta\cos\theta \\
\theta'' &= \frac{g\sin\theta(M+m) - F(t)\cos\theta + kx\cos\theta - mL(\theta')^2\sin\theta\cos\theta}{L(M+m-m\cos^2\theta)} \qquad (4)
\end{aligned}$$

Eqs. (3) and (4) is what we will use to convert the system to first order form since they are in decoupled form.

Convert to state space form

Using the decoupled ODE's above, Eqs (3) and (4), and introducing 4 state variables $x_1, x_2, x_3\, x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

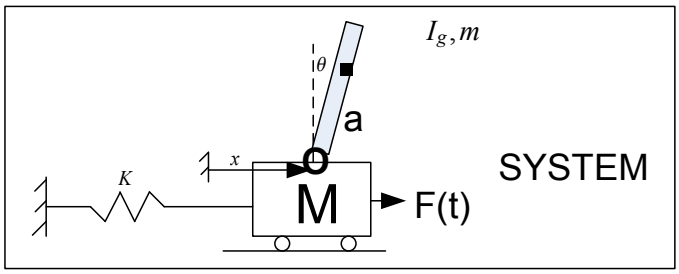
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t)-kx-mg\sin\theta\cos\theta+mL(\theta')^2\sin\theta}{(M+m-m\cos^2\theta)} \\ \frac{g\sin\theta(M+m)-F(t)\cos\theta+kx\cos\theta-mL(\theta')^2\sin\theta\cos\theta}{L(M+m-m\cos^2\theta)} \end{pmatrix}$$

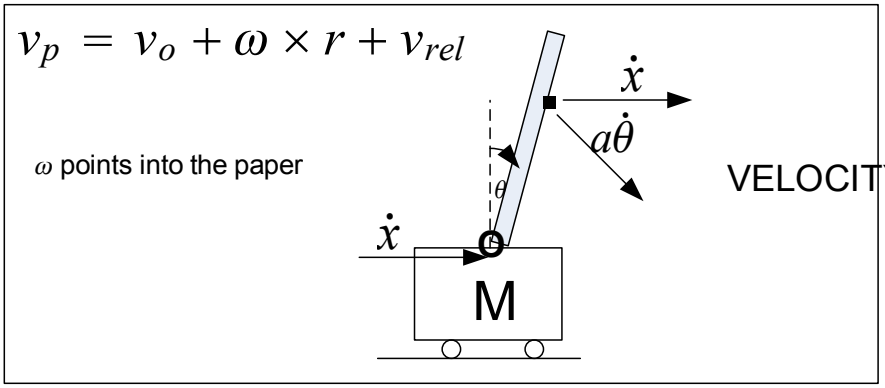
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{F(t)-kx_1-mg\sin x_2\cos x_2+mLx_4^2\sin x_1}{(M+m-m\cos^2 x_2)} \\ \frac{g\sin x_2(M+m)-F(t)\cos x_2+kx_1\cos x_2-mLx_4^2\sin x_2\cos x_2}{L(M+m-m\cos^2 x_2)} \end{pmatrix}$$

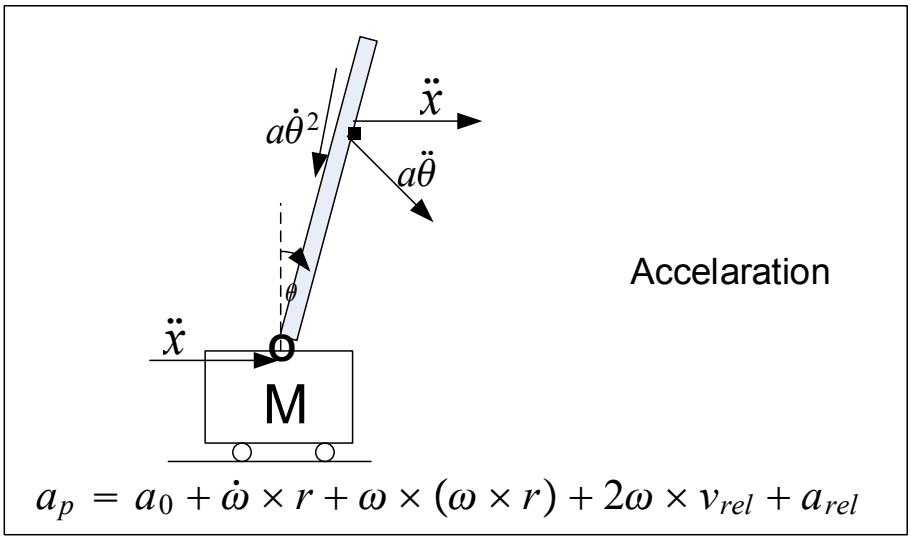
3.2.3 problem 2

Velocity diagram

Only the velocity diagram is needed for the Lagrangian method. The generalized coordinates are: x, θ







Let L be the Lagrangian, and let T be the kinetic energy of the system, and V the potential energy.

$$L = T - V$$

$$T = \frac{1}{2}M\dot{x}^2 + \overbrace{\frac{1}{2}I_g\dot{\theta}^2}^{\text{angular bar K.E.}} + \overbrace{\frac{1}{2}m\left(\dot{x}^2 + \left(a\dot{\theta}\right)^2 + 2\dot{x}a\dot{\theta}\cos\theta\right)}^{\text{linear bar K.E. of its c.g.}}$$

And

$$V = \frac{1}{2}kx^2 + mga\cos\theta$$

Therefore, L becomes

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}I_g\dot{\theta}^2 + \frac{1}{2}m\left(\dot{x}^2 + \left(a\dot{\theta}\right)^2 + 2\dot{x}a\dot{\theta}\cos\theta\right) - \left(\frac{1}{2}kx^2 + mga\cos\theta\right) \end{aligned}$$

To obtain equations of motion, we evaluate

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$$

For M we obtain

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} + ma\dot{\theta}\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = M\ddot{x} + m\ddot{x} + ma\ddot{\theta}\cos\theta - ma\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x} = -kx$$

Hence, for the mass M, the equation of motion is

$$M\ddot{x} + m\ddot{x} + ma\ddot{\theta}\cos\theta - ma\dot{\theta}^2\sin\theta + kx = F$$

$$\ddot{x} = \frac{F - kx - ma\ddot{\theta}\cos\theta + ma\dot{\theta}^2\sin\theta}{M + m} \tag{1}$$

To find EQM for mass m, evaluate $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$

$$\frac{\partial L}{\partial \dot{\theta}} = I_g\dot{\theta} + m\left(a^2\dot{\theta} + \dot{x}a\cos\theta\right)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = I_g\ddot{\theta} + m\left(a^2\ddot{\theta} - \dot{x}a\dot{\theta}\sin\theta + \ddot{x}a\cos\theta\right)$$

$$\frac{\partial L}{\partial \theta} = -m\dot{x}a\dot{\theta}\sin\theta + mga\sin\theta$$

Hence EQM is

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} &= 0 \\ I_g\ddot{\theta} + m\left(a^2\ddot{\theta} - \dot{x}a\dot{\theta}\sin\theta + \ddot{x}a\cos\theta\right) + m\dot{x}a\dot{\theta}\sin\theta - mga\sin\theta &= 0 \\ I_g\ddot{\theta} + ma^2\ddot{\theta} - m\dot{x}a\dot{\theta}\sin\theta + m\ddot{x}a\cos\theta + m\dot{x}a\dot{\theta}\sin\theta - mga\sin\theta &= 0 \\ I_g\ddot{\theta} + ma^2\ddot{\theta} + m\ddot{x}a\cos\theta - mga\sin\theta &= 0 \\ \ddot{\theta}\left(I_g + ma^2\right) &= mga\sin\theta - m\ddot{x}a\cos\theta \\ \ddot{\theta} &= \frac{mga\sin\theta - m\ddot{x}a\cos\theta}{I_g + ma^2} \\ &= \frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g} \end{aligned} \tag{2}$$

Decouple the ODE's

Substitute Eq. (2) into Eq. (1) gives

$$x'' = \frac{F(t) - kx - ma\left[\frac{mag\sin\theta - max''\cos\theta}{ma^2 + I_g}\right]\cos\theta + ma(\theta')^2\sin\theta}{M+m}$$

Let $\left(ma^2 + I_g\right) = I_o$, hence

$$x'' = \frac{I_o F(t) - I_o kx - ma\left[mag\sin\theta - max''\cos\theta\right]\cos\theta + I_o ma(\theta')^2\sin\theta}{I_o(M+m)}$$

$$I_o(M+m)x'' - m^2a^2x''\cos\theta = I_o F(t) - I_o kx - m^2a^2g\sin\theta\cos\theta + I_o ma(\theta')^2\sin\theta$$

$$x'' = \frac{I_o F(t) - I_o kx - m^2a^2g\sin\theta\cos\theta + I_o ma(\theta')^2\sin\theta}{I_o(M+m) - m^2a^2\cos\theta} \tag{3}$$

Also, we can solve for θ'' by Substituting Eq. (1) into (2)

$$\theta'' = \frac{mag\sin\theta - ma\left[\frac{F(t) - kx - ma\theta''\cos\theta + ma(\theta')^2\sin\theta}{M+m}\right]\cos\theta}{ma^2 + I_g}$$

Let $\left(ma^2 + I_g\right) = I_o$, hence

$$\theta'' I_o(M+m) = (M+m)mag\sin\theta - ma\left[F(t) - kx - ma\theta''\cos\theta + ma(\theta')^2\sin\theta\right]\cos\theta$$

$$= (M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta + m^2a^2\theta''\cos^2\theta - m^2a^2(\theta')^2\sin\theta\cos\theta$$

$$\theta''\left[I_o(M+m) - m^2a^2\cos^2\theta\right] = (M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta$$

$$\theta'' = \frac{(M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta}{I_o(M+m) - m^2a^2\cos^2\theta} \tag{4}$$

Therefore, the final EQM are

$$x'' = \frac{I_o F(t) - I_o kx - m^2a^2g\sin\theta\cos\theta + I_o ma(\theta')^2\sin\theta}{I_o(M+m) - m^2a^2\cos\theta}$$

$$\theta'' = \frac{(M+m)mag\sin\theta - maF(t)\cos\theta + makx\cos\theta - m^2a^2(\theta')^2\sin\theta\cos\theta}{I_o(M+m) - m^2a^2\cos^2\theta}$$

Convert to state space form

Using the decoupled ODE's above, Eqs (3) and (4), and introducing 4 state variables $x_1, x_2, x_3\, x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

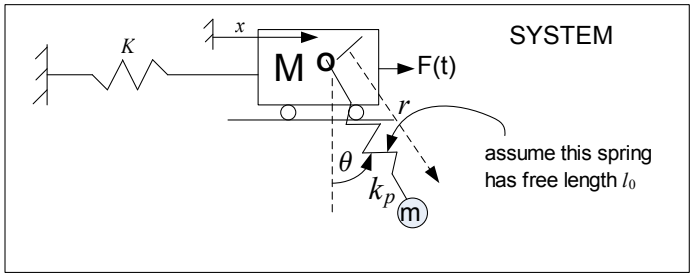
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) - I_o k x - m^2 a^2 g \sin\theta \cos\theta + I_o m a (\theta')^2 \sin\theta}{I_o(M+m) - m^2 a^2 \cos\theta} \\ \frac{(M+m) m a g \sin\theta - m a F(t) \cos\theta + m a k x \cos\theta - m^2 a^2 (\theta')^2 \sin\theta \cos\theta}{I_o(M+m) - m^2 a^2 \cos^2\theta} \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) - I_o k x_1 - m^2 a^2 g \sin x_2 \cos x_2 + I_o m a\ x_4^2 \sin x_2}{I_o(M+m) - m^2 a^2 \cos x_2} \\ \frac{(M+m) m a g \sin x_2 - m a F(t) \cos x_2 + m a k x \cos x_2 - m^2 a^2 x_4^2 \sin x_2 \cos x_2}{I_o(M+m) - m^2 a^2 \cos^2 x_2} \end{pmatrix}$$

3.2.4 Solution problem 3

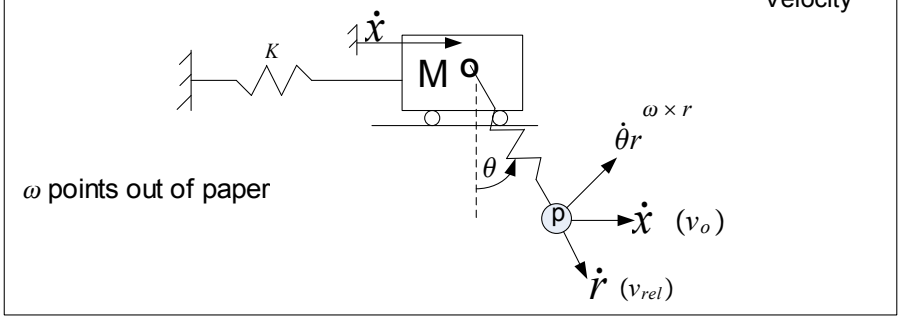
Velocity diagram

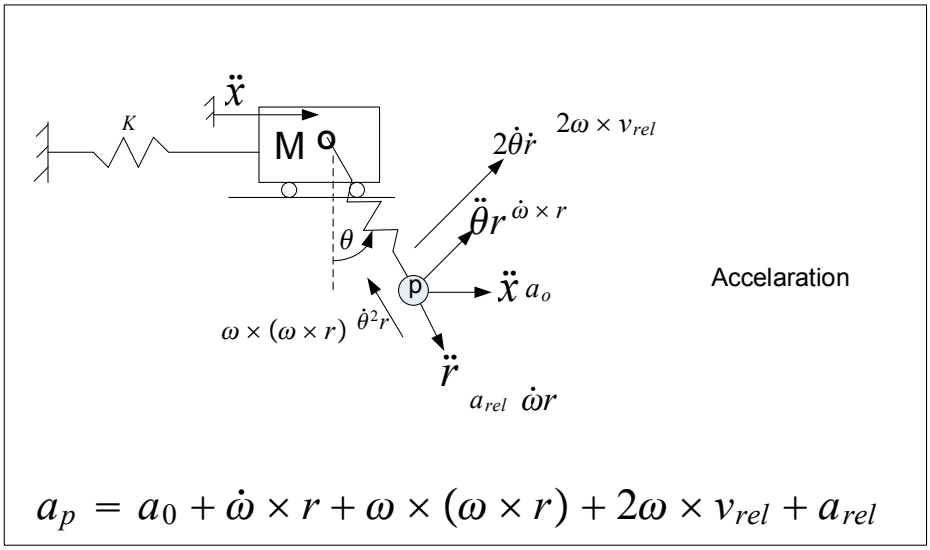
Only the velocity diagram is needed for the Lagrangian method. Generalized coordinates are: x, θ, r



$$v_p = v_o + \omega \times r + v_{rel}$$

Velocity





$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$

Let L be the Lagrangian, and let T be the kinetic energy of the system, and V the potential energy.

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\left(\dot{x}\cos\theta + \dot{\theta}r\right)^2 + (\dot{x}\sin\theta + \dot{r})^2\right)$$

$$V = \frac{1}{2}kx^2 + \frac{1}{2}k_p\,(r - l_0)^2 - mgr\cos\theta$$

Hence

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\left(\dot{x}^2\cos^2\theta + \dot{\theta}^2r^2 + 2\dot{x}\dot{\theta}r\cos\theta\right) + \left(\dot{x}^2\sin^2\theta + \dot{r}^2 + 2\dot{r}\dot{x}\sin\theta\right)\right) \\ &- \left(\frac{1}{2}kx^2 + \frac{1}{2}k_p\left(r - l_0\right)^2 - mgr\cos\theta\right) \end{aligned}$$

$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + \dot{\theta}^2r^2 + 2\dot{x}\dot{\theta}r\cos\theta + \dot{r}^2 + 2\dot{r}\dot{x}\sin\theta\right) - \frac{1}{2}kx^2 - \frac{1}{2}k_p\left(r - l_0\right)^2 + mgr\cos\theta$$

For x we obtain

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = Q_x$$

$$kx + m\sin\theta\left(-r\dot{\theta}^2 + \ddot{r}\right) + (m+M)\ddot{x} + m\cos\theta\left(2\dot{r}\dot{\theta} + r\ddot{\theta}\right) = Q_x$$

To find Q_x we make small δx displacement and find the virtual work done. Hence we obtain

$$\delta W = F\delta x$$

Hence $Q_x = F(t)$ and EQM for x becomes

$$kx + m\sin\theta\left(-r\dot{\theta}^2 + \ddot{r}\right) + (m+M)\ddot{x} + m\cos\theta\left(2\dot{r}\dot{\theta} + r\ddot{\theta}\right) = F$$

Hence

$$\ddot{x} = \frac{F - kx - m\sin\theta\left(-r\dot{\theta}^2 + \ddot{r}\right) - m\cos\theta\left(2\dot{r}\dot{\theta} + r\ddot{\theta}\right)}{(m+M)} \tag{1}$$

For r we obtain

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{r}} - \frac{\partial L}{\partial r} = Q_r$$

$$-k_pl_0 - gm\cos\theta + r\left(k_p - m\dot{\theta}^2\right) + m\ddot{r} + m\ddot{x}\sin\theta = Q_r$$

To find Q_r we make small δr displacement and find the virtual work done. Hence we obtain $Q_r = 0$ and EQM for r becomes

$$-k_pl_0 - gm\cos\theta + r\left(k_p - m\dot{\theta}^2\right) + m\ddot{r} + m\ddot{x}\sin\theta = 0$$

Hence

$$\ddot{r} = \frac{k_pl_0 + gm\cos\theta - r\left(k_p - m\dot{\theta}^2\right) - m\ddot{x}\sin\theta}{m} \tag{2}$$

For θ we obtain

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q_\theta$$

$$mr\left(g\sin\theta + 2\dot{r}\dot{\theta} + \cos\theta\ddot{x} + r\ddot{\theta}\right) = Q_\theta$$

To find Q_θ we make small $\delta\theta$ displacement and find the virtual work done. Hence we obtain $Q_\theta = 0$ and EQM for θ becomes

$$mr\left(g\sin\theta + 2\dot{r}\dot{\theta} + \cos\theta\ddot{x} + r\ddot{\theta}\right) = 0$$

Hence

$$\ddot{\theta} = \frac{-g\sin\theta - 2\dot{r}\dot{\theta} - \ddot{x}\cos\theta}{r} \tag{3}$$

To be able to convert to first order form, we need to decouple the above equations. This results in

$$\begin{aligned}
\ddot{x} &= \frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta \\
\ddot{r} &= g\cos\theta - \frac{k_p}{m}(r - l_0) - \frac{F(t)}{M}\sin\theta + \frac{k}{M}x\sin\theta + \frac{k_p}{M}(r - l_0)\sin^2\theta + \dot{\theta}^2 r \\
\ddot{\theta} &= -\frac{g}{r}\sin\theta - \frac{2\dot{r}}{r}\dot{\theta} - \frac{F(t)}{rM}\cos\theta + \frac{k}{rM}x\cos\theta + \frac{k_p}{rM}(r - l_0)\sin\theta\cos\theta
\end{aligned}$$

Convert to state space form

Using the decoupled ODE's above, and introducing 6 state variables $x_1, x_2, x_3, x_4, x_5, x_6$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ r \\ x' \\ \theta' \\ r' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ r' \\ x'' \\ \theta'' \\ r'' \end{pmatrix}$$

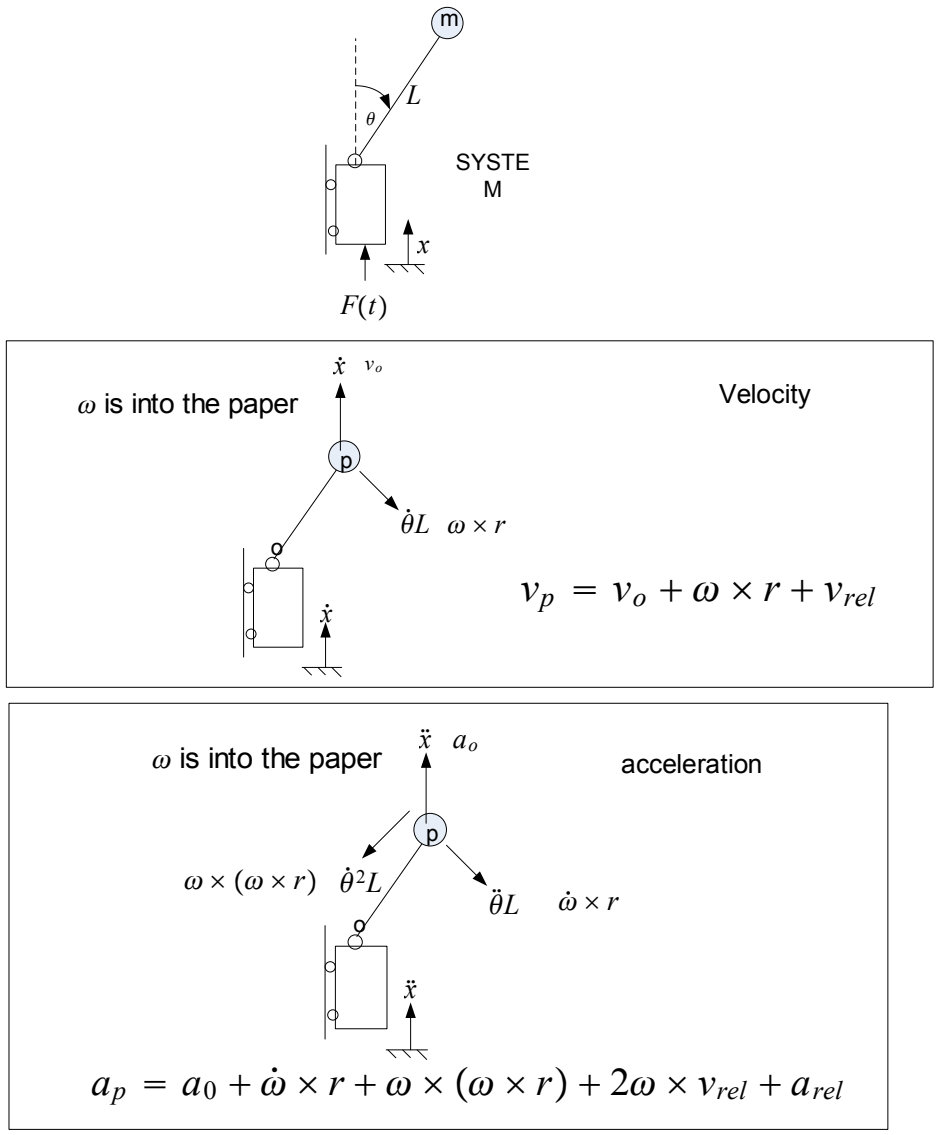
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{F(t)}{M} - \frac{k}{M}x - \frac{k_p}{M}(r - l_0)\sin\theta \\ -\frac{g}{r}\sin\theta - \frac{2r'}{r}\theta' - \frac{F(t)}{rM}\cos\theta + \frac{k}{rM}x\cos\theta + \frac{k_p}{rM}(r - l_0)\sin\theta\cos\theta \\ g\cos\theta - \frac{k_p}{m}(r - l_0) - \frac{F(t)}{M}\sin\theta + \frac{k}{M}x\sin\theta + \frac{k_p}{M}(r - l_0)\sin^2\theta + \dot{\theta}^2 r \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{F(t)}{M} - \frac{k}{M}x_1 - \frac{k_p}{M}(x_3 - l_0)\sin x_2 \\ -\frac{g}{r}\sin x_2 - \frac{2x_6}{x_3}x_5 - \frac{F(t)}{x_3 M}\cos x_2 + \frac{k}{x_3 M}x_1\cos x_2 + \frac{k_p}{x_3 M}(x_3 - l_0)\sin x_2\cos x_2 \\ g\cos x_2 - \frac{k_p}{m}(x_3 - l_0) - \frac{F(t)}{M}\sin x_2 + \frac{k}{M}x_1\sin x_2 + \frac{k_p}{M}(x_3 - l_0)\sin^2 x_2 + x_5^2 x_3 \end{pmatrix}$$

3.2.5 problem 4

Velocity Diagram

Only the velocity diagram is needed for the Lagrangian method. Generalized coordinates are: x, θ



Let L be the Lagrangian, and let T be the kinetic energy of the system, and V the potential energy.

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\left(\dot{x} - L\dot{\theta}\right)^2 + \left(L\dot{\theta}\cos\theta\right)^2\right)$$
$$V = Mgx + mg\left(L\cos\theta + x\right)$$

Hence

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + L^2\dot{\theta}^2 - 2\dot{x}L\dot{\theta} + L^2\dot{\theta}^2\cos^2\theta\right) - Mgx - mg\left(L\cos\theta + x\right) \end{aligned}$$

For x we obtain

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = Q_x$$
$$g\left(m + M\right) + \left(m + M\right)\ddot{x} - Lm\ddot{\theta} = Q_x$$

To find Q_x we make small δx displacement and find the virtual work done. Hence we obtain

$$\delta W = F\delta x$$

Therefore $Q_x = F$ and the EQM for x becomes

$$g\left(m+M\right)+\left(m+M\right)\ddot{x}-Lm\ddot{\theta}=F$$

Hence

$$\ddot{x}=\frac{F-g\left(m+M\right)+Lm\ddot{\theta}}{m+M} \tag{1}$$

For θ we have

$$\begin{aligned}\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}}-\frac{\partial L}{\partial\theta}&=Q_\theta\\ Lm\left(g\sin\theta+L\dot{\theta}^2\cos\theta\sin\theta+\ddot{x}-L\ddot{\theta}\left(1+\cos^2\theta\right)\right)&=Q_\theta\end{aligned}$$

To find Q_θ we make small $\delta\theta$ displacement and find the virtual work done. We find that $Q_\theta = 0$ hence the EQM for θ becomes

$$\begin{aligned}Lm\left(g\sin\theta+L\dot{\theta}^2\cos\theta\sin\theta+\ddot{x}-L\ddot{\theta}\left(1+\cos^2\theta\right)\right)&=0\\ g\sin\theta+L\dot{\theta}^2\cos\theta\sin\theta+\ddot{x}-L\ddot{\theta}\left(1+\cos^2\theta\right)&=0\end{aligned}$$

Hence

$$\ddot{\theta}=\frac{g\sin\theta+L\dot{\theta}^2\cos\theta\sin\theta+\ddot{x}}{L\left(1+\cos^2\theta\right)} \tag{2}$$

To convert to first form, Eqs. (1) and (2) are decoupled resulting in

$$\begin{aligned}\ddot{x}&=\frac{\left(F-g\left(m+M\right)\right)\left(1+\cos^2\theta\right)+mg\sin\theta+Lm\dot{\theta}^2\cos\theta\sin\theta}{M+\left(M+m\right)\cos^2\theta}\\ \ddot{\theta}&=\frac{F-g\left(m+M\right)+g\left(m+M\right)\sin\theta+L\left(m+M\right)\dot{\theta}^2\cos\theta\sin\theta}{L\left(M+\left(M+m\right)\cos^2\theta\right)}\end{aligned}$$

Convert to state space form

Using the decoupled ODE's above, and introducing 4 state variables x_1, x_2, x_3, x_4 we obtain

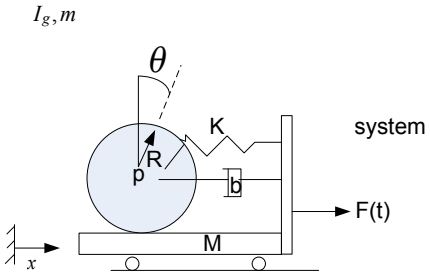
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{(F-g(m+M))\left(1+\cos^2\theta\right)+mg\sin\theta+Lm\dot{\theta}^2\cos\theta\sin\theta}{M+(M+m)\cos^2\theta} \\ \frac{F-g(m+M)+g(m+M)\sin\theta+L(m+M)\dot{\theta}^2\cos\theta\sin\theta}{L(M+(M+m)\cos^2\theta)} \end{pmatrix}$$

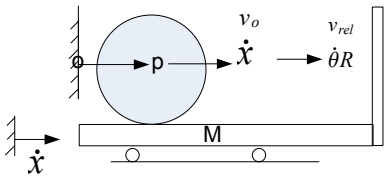
$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{(F-g(m+M))\left(1+\cos^2 x_2\right)+mg\sin x_2+Lmx_4^2\cos x_2\sin x_2}{M+(M+m)\cos^2 x_2} \\ \frac{F-g(m+M)+g(m+M)\sin x_2+L(m+M)x_4^2\cos x_2\sin x_2}{L(M+(M+m)\cos^2 x_2)} \end{pmatrix}$$

3.2.6 Solution problem 5

Only the velocity diagram is needed for the Lagrangian method.

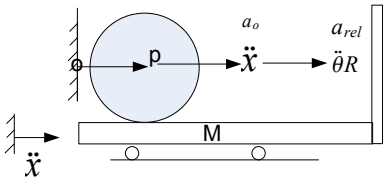


Velocity

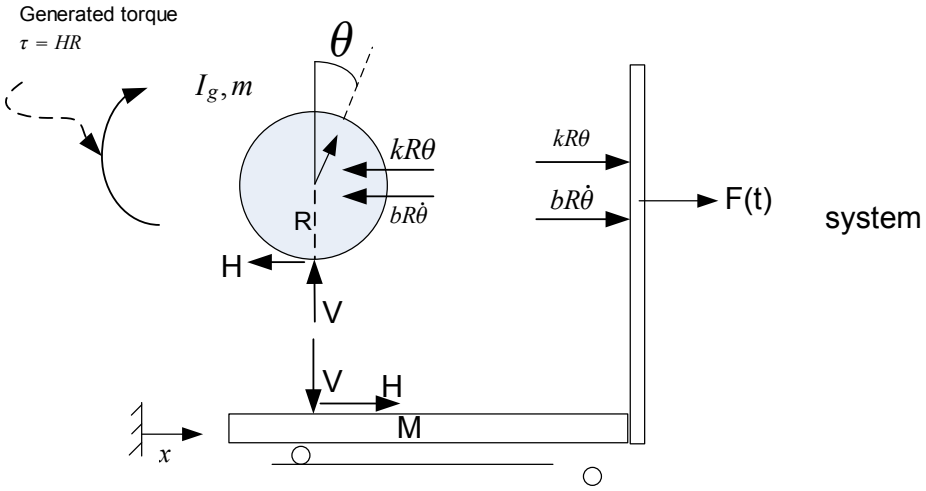


$$v_p = v_o + \omega \times r + v_{rel}$$

acceleration



$$a_p = a_0 + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$



Let L be the Lagrangian, and let T be the kinetic energy of the system, and V the potential energy.

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x} + R\dot{\theta}\right)^2 + \frac{1}{2}I_g\dot{\theta}^2$$

$$V = \frac{1}{2}k\left(R\theta\right)^2$$

Hence

$$\begin{aligned} L &= T - V \\ &= \left[\frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x} + R\dot{\theta}\right)^2 + \frac{1}{2}I_g\dot{\theta}^2\right] - \frac{1}{2}k\left(R\theta\right)^2 \\ &= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + R^2\dot{\theta}^2 + 2\dot{x}\dot{\theta}R\right) + \frac{1}{2}I_g\dot{\theta}^2 - \frac{1}{2}kR^2\theta^2 \end{aligned}$$

and

$$\begin{aligned} \frac{\partial L}{\partial \dot{x}} &= M\dot{x} + m\dot{x} + m\dot{\theta}R \\ \frac{\partial L}{\partial x} &= 0 \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= M\ddot{x} + m\ddot{x} + m\ddot{\theta}R \end{aligned}$$

Hence, for x, the EQM is

$$M\ddot{x} + m\ddot{x} + m\ddot{\theta}R = Q_x$$

Where Q_x is the generalized force which is

$$\begin{aligned} \delta W &= \frac{F\delta x + \left(kR\theta\right)\delta x + \left(bR\dot{\theta}\right)\delta x}{\delta x} \\ &= F + bR\dot{\theta} + kR\theta \end{aligned}$$

hence

$$\begin{aligned} M\ddot{x} + m\ddot{x} + m\ddot{\theta}R &= F\left(t\right) + bR\dot{\theta} + kR\theta \\ \ddot{x} &= \frac{F\left(t\right) + bR\dot{\theta} + kR\theta - m\ddot{\theta}R}{M + m} \end{aligned}$$

Since $I_g = \frac{mR^2}{2}$, then the above can be written as

$$\ddot{x} = \frac{RF\left(t\right) + bR^2\dot{\theta} + kR^2\theta - 2I_g\ddot{\theta}}{R\left(M + m\right)}$$

For θ, the EQM is

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}} &= m\left(R^2\dot{\theta} + \dot{x}R\right) + I_g\dot{\theta} \\ \frac{\partial L}{\partial \theta} &= -kR^2\theta \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} \end{aligned}$$

Hence

$$m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} + kR^2\theta = F_\theta$$

In this case, $\delta W = -\frac{\left(bR\dot{\theta}\right)R\delta\theta}{\delta\theta}$, hence $F_\theta = -bR^2\dot{\theta}$, therefore, the EQM is

$$\begin{aligned} m\left(R^2\ddot{\theta} + \ddot{x}R\right) + I_g\ddot{\theta} + kR^2\theta &= -bR^2\dot{\theta} \\ \ddot{\theta} &= \frac{-bR^2\dot{\theta} - kR^2\theta - m\ddot{x}R}{I_g + mR^2} \\ &= -\frac{bR^2\dot{\theta} + kR^2\theta + m\ddot{x}R}{I_o} \end{aligned}$$

By decoupling the 2 equations of motion we obtain

$$x'' = \frac{I_o F(t) + kI_o R\theta + bI_o R\theta' + I_g\left[kR\theta + bR\theta'\right]}{MI_o - I_g m}$$

$$\theta'' = \frac{-mRF(t) - mkR^2\theta - mbR^2\theta' - MkR^2\theta - MbR^2\theta'}{MI_o - I_g m}$$

Convert to state space form

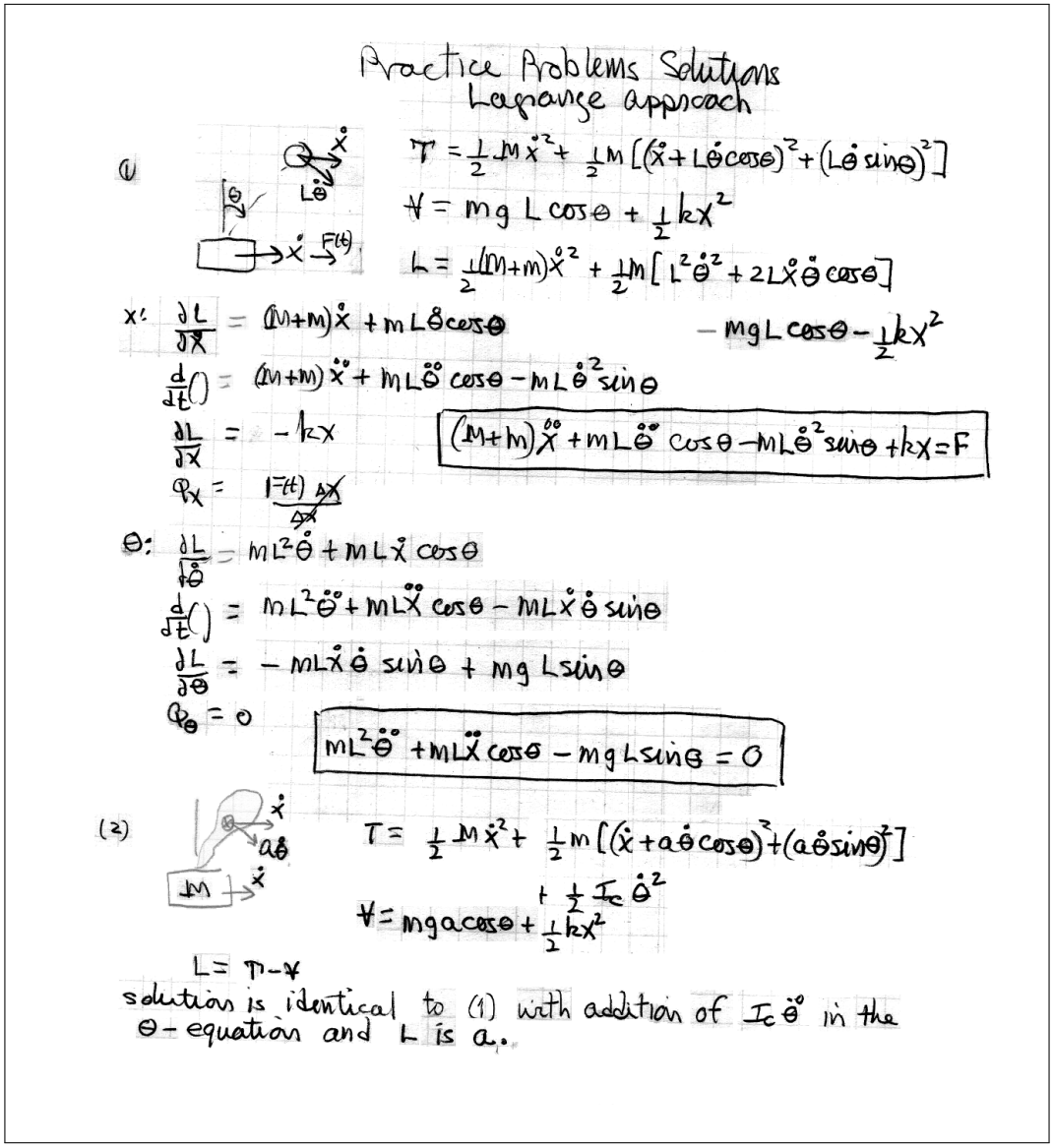
Using the decoupled ODE's above, and introducing 4 state variables x_1, x_2, x_3, x_4 we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \theta \\ x' \\ \theta' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x' \\ \theta' \\ x'' \\ \theta'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) + kI_o R\theta + bI_o R\theta' + I_g[kR\theta + bR\theta']}{MI_o - I_g m} \\ \frac{-mRF(t) - mkR^2\theta - mbR^2\theta' - MkR^2\theta - MbR^2\theta'}{MI_o - I_g m} \end{pmatrix}$$

$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{I_o F(t) + kI_o Rx_2 + bI_o Rx_4 + I_g[kRx_2 + bRx_4]}{MI_o - I_g m} \\ \frac{-mRF(t) - mkR^2x_2 - mbR^2x_4 - MkR^2x_2 - MbR^2x_4}{MI_o - I_g m} \end{pmatrix}$$

3.2.7 Key solution



Practice Problems Solutions
Lagrange approach

① $T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[(\dot{x}+L\dot{\theta}\cos\theta)^2 + (L\dot{\theta}\sin\theta)^2\right]$

$V = mgL\cos\theta + \frac{1}{2}kx^2$

$L = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}m\left[L^2\dot{\theta}^2 + 2L\dot{x}\dot{\theta}\cos\theta\right]$

x: $\frac{\partial L}{\partial x} = (M+m)\dot{x} + mL\dot{\theta}\cos\theta \qquad -mgL\cos\theta - \frac{1}{2}kx^2$

$\frac{d}{dt}() = (M+m)\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta$

$\frac{\partial L}{\partial x} = -kx$ $\qquad \boxed{(M+m)\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta + kx = F}$

$Q_x = F(t)\,\Delta x$

θ: $\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} + mL\dot{x}\cos\theta$

$\frac{d}{dt}() = mL^2\ddot{\theta} + mL\ddot{x}\cos\theta - mL\dot{x}\dot{\theta}\sin\theta$

$\frac{\partial L}{\partial \theta} = -mL\dot{x}\dot{\theta}\sin\theta + mgL\sin\theta$

$Q_\theta = 0$

$\boxed{mL^2\ddot{\theta} + mL\ddot{x}\cos\theta - mgL\sin\theta = 0}$

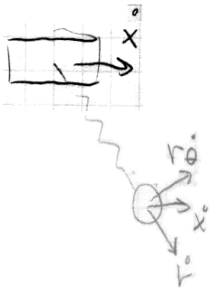
(2) $T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[(\dot{x}+a\dot{\theta}\cos\theta)^2 + (a\dot{\theta}\sin\theta)^2\right] + \frac{1}{2}I_c\dot{\theta}^2$

$V = mga\cos\theta + \frac{1}{2}kx^2$

$L = T - V$

solution is identical to (1) with addition of $I_c\ddot{\theta}$ in the θ-equation and L is a.

(3)



$$T = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\left[(\dot{r}+\dot{x}\sin\theta)^2 + (r\dot{\theta}+\dot{x}\cos\theta)^2\right]$$

$$V = \frac{1}{2}kx^2 + \frac{1}{2}k_p(r-l_0)^2 - mgr\cos\theta$$

$$L = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\left[\dot{r}^2 + \dot{x}^2 + r^2\dot{\theta}^2 + 2\dot{x}\dot{r}\sin\theta + 2r\dot{x}\dot{\theta}\cos\theta\right] - \frac{1}{2}kx^2 - \frac{1}{2}k_p(r-l_0)^2 + mgr\cos\theta$$

x: $\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} + m\dot{r}\sin\theta + mr\dot{\theta}\cos\theta$

$\frac{d}{dt}(\,) = (M+m)\ddot{x} + m\ddot{r}\sin\theta + m\dot{r}\dot{\theta}\cos\theta + m\dot{r}\dot{\theta}\cos\theta + mr\ddot{\theta}\cos\theta - mr\dot{\theta}^2\sin\theta$

$\frac{\partial L}{\partial x} = -kx$

$Q_x = F\frac{\Delta x}{\Delta x}$

$$(M+m)\ddot{x} + m\ddot{r}\sin\theta + 2m\dot{r}\dot{\theta}\cos\theta + mr\ddot{\theta}\cos\theta - mr\dot{\theta}^2\sin\theta + kx = F$$

r: $\frac{\partial L}{\partial \dot{r}} = m\dot{r} + m\dot{x}\sin\theta$

$\frac{d}{dt}(\,) = m\ddot{r} + m\ddot{x}\sin\theta + m\dot{x}\dot{\theta}\cos\theta$

$\frac{\partial L}{\partial r} = mr\dot{\theta}^2 + m\dot{x}\dot{\theta}\cos\theta - k_p(r-l_0) + mg\cos\theta$

$Q_r = 0$

$$m\ddot{r} + m\ddot{x}\sin\theta - mr\dot{\theta}^2 + k_p(r-l_0) - mg\cos\theta = 0$$

θ: $\frac{\partial L}{\partial \dot{\theta}} = mr^2\dot{\theta} + mr\dot{x}\cos\theta$

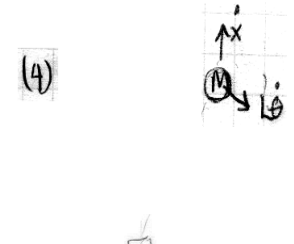
$\frac{d}{dt}(\,) = 2mr\dot{r}\dot{\theta} + mr^2\ddot{\theta} + m\dot{r}\dot{x}\cos\theta + mr\ddot{x}\cos\theta - mr\dot{x}\dot{\theta}\sin\theta$

$\frac{\partial L}{\partial \theta} = m\dot{x}\dot{r}\cos\theta - mr\dot{x}\dot{\theta}\sin\theta - mgr\sin\theta$

$Q_\theta = 0$

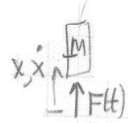
$$mr^2\ddot{\theta} + mr\ddot{x}\cos\theta + 2mr\dot{r}\dot{\theta} + mgr\sin\theta = 0$$

(4)



$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[(\dot{x} - L\dot{\theta}\sin\theta)^2 + (L\dot{\theta}\cos\theta)^2\right]$$

$$V = Mgx + mg[x + L\cos\theta]$$



$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[\dot{x}^2 + L^2\dot{\theta}^2 - 2L\dot{x}\dot{\theta}\sin\theta\right] - Mgx - mg[x + L\cos\theta]$$

x:

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m\dot{x} - mL\dot{\theta}\sin\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = (m+M)\ddot{x} - mL\ddot{\theta}\sin\theta - mL\dot{\theta}^2\cos\theta$$

$$\frac{\partial L}{\partial x} = -Mg - mg$$

$$Q_x = F\frac{\Delta x}{\Delta x}$$

$$\boxed{(m+M)\ddot{x} - mL\ddot{\theta}\sin\theta - mL\dot{\theta}^2\cos\theta + (m+M)g = F}$$

θ:

$$\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} - mL\dot{x}\sin\theta$$

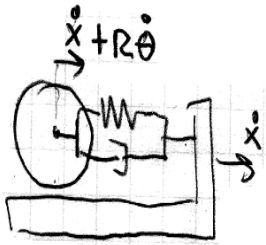
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = mL^2\ddot{\theta} - mL\ddot{x}\sin\theta - mL\dot{x}\dot{\theta}\cos\theta$$

$$\frac{\partial L}{\partial \theta} = -mL\dot{x}\dot{\theta}\cos\theta + mgL\sin\theta$$

$$Q_\theta = 0$$

$$\boxed{mL^2\ddot{\theta} - mL\ddot{x}\sin\theta - mgL\sin\theta = 0}$$

15)



$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[(\dot{x}+R\dot{\theta})^2\right] + \frac{1}{2}I_g\dot{\theta}^2$$

$$V = \frac{1}{2}k(R\theta)^2$$

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[\dot{x}^2 + R^2\dot{\theta}^2 + 2R\dot{x}\dot{\theta}\right] + \frac{1}{2}I_g\dot{\theta}^2 - \frac{1}{2}kR^2\theta^2$$

x:

$$\frac{\partial L}{\partial \dot{x}} = (M+m)\dot{x} + mR\dot{\theta}$$

$$\frac{d}{dt}() = (M+m)\ddot{x} + mR\ddot{\theta}$$

$$\frac{\partial L}{\partial x} = 0$$

$$\boxed{(M+m)\ddot{x} + mR\ddot{\theta} = F}$$

$$Q_x = F\frac{\Delta x}{\Delta x}$$

θ

$$\frac{\partial L}{\partial \dot{\theta}} = mR^2\dot{\theta} + mR\dot{x} + I_g\dot{\theta}$$

$$\frac{d}{dt}() = (I_g + mR^2)\ddot{\theta} + mR\ddot{x}$$

$$\frac{\partial L}{\partial \theta} = -kR^2\theta$$

$$Q_\theta = -b(R\dot{\theta})\frac{R\Delta\theta}{\Delta\theta} = -bR^2\dot{\theta}$$

$$\boxed{(I_g + mR^2)\ddot{\theta} + mR\ddot{x} + kR^2\theta + bR^2\dot{\theta} = 0}$$

3.3 Problem 7.2 part (e) in Textbook

Starting with

$$q'(t) + \frac{(I_1 - I_3)}{I_2}\Omega\, r(t) = 0 \tag{1}$$

$$r'(t) + \frac{(I_2 - I_1)}{I_3}\Omega\, q(t) = 0 \tag{2}$$

To decouple the ODE's, take derivatives and substituting back, we find

$$q''(t) + k\, q(t) = 0 \tag{3}$$

$$r''(t) + k\, r(t) = 0 \tag{4}$$

Where $k = \frac{(I_1 - I_3)(I_2 - I_2)}{I_2 I_3}\Omega^2$. The solution to the above is (for stability $k > 0$)

$$q(t) = A\cos\sqrt{k}t + B\sin\sqrt{k}t \tag{5}$$

$$r(t) = C\cos\sqrt{k}t + D\sin\sqrt{k}t \tag{6}$$

Now, $q(0) = \frac{\text{Imp}}{I_{yy}}$, hence $A = \frac{\text{Imp}}{I_{yy}}$. To find B take derivative

$$q'(t) = -\sqrt{k}\frac{\text{Imp}}{I_{yy}}\sin\sqrt{k}t + B\sqrt{k}\cos\sqrt{k}t \tag{7}$$

But at $t = 0$ then we go back and use Eq. (1) to find $q'(0)$, and equate the result to the above at $t = 0$. Eq (1), at $t = 0$, gives

$$q'(0) + \frac{(I_1 - I_3)}{I_2}\Omega\, r(0) = 0$$

But $r(0) = 0$, hence $q'(0) = 0$, and so this results in $B = 0$ in Eq.(7), hence the solution for $q(t)$ is

$$q(t) = \frac{\text{Imp}}{I_{yy}}\cos\sqrt{k}t$$

We do the same for $r(t)$. From Eq. (6) we find that $C = 0$ since $r(0) = 0$, so now $r(t)$ reduces to

$$r(t) = D\sin\sqrt{k}t$$

Hence

$$r'(t) = \sqrt{k}D\cos\sqrt{k}t \tag{8}$$

To find D, then we go back and use Eq. (2) to find $r'(0)$, and equate the result to the above at $t = 0$. Eq (2), at $t = 0$, gives

$$r'(0) + \frac{(I_2 - I_1)}{I_3}\Omega\, q(0) = 0$$

But $q(0) = \frac{\text{Imp}}{I_{yy}}$, hence

$$r'(0) = -\frac{(I_2 - I_1)}{I_3}\Omega\,\frac{\text{Imp}}{I_{yy}}$$

Therefore, equate the above to Eq. (8) evaluated at $t = 0$ gives

$$D = -\frac{(I_2 - I_1)}{I_3\sqrt{k}}\Omega\,\frac{\text{Imp}}{I_{yy}}$$

Hence Eq. (6) becomes

$$r(t) = \left(-\frac{(I_2 - I_1)}{I_3\sqrt{k}}\Omega\,\frac{\text{Imp}}{I_{yy}}\right)\sin\sqrt{k}t$$

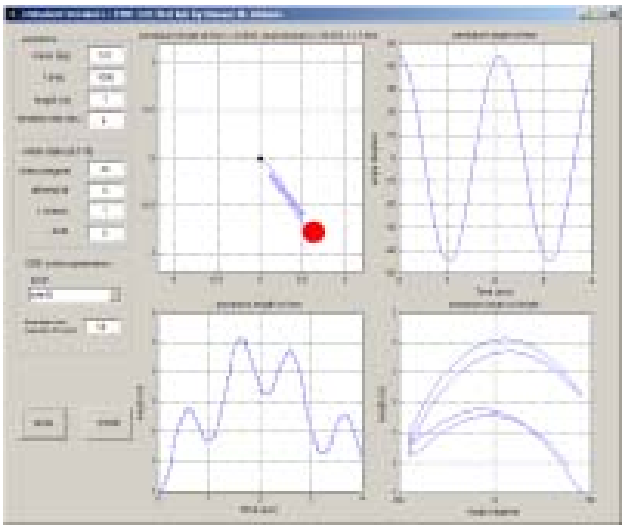
so $r(t)$ is sinusoidal as well.

Chapter 4

Lab projects

There was 5 lab assignments and 2 extra problem sets.

4.1 Lab 1, Simulate spring pendulum. GUI application using Matlab



4.1.1 Problem description

We are asked to write a program to simulation the equation of motion of the following spring pendulum shown in the textbook on page 42

4.1.2 Code

Here is 45 MB AVI movie showing the program I wrote to solve this problem. pendulum.avi

To run the program, download this code.zip and extract it. It will create a new folder called code

Next, add this folder to your Matlab path (using Matlab `file-> set path`).

Then from Matlab command window, type `eme_121_lab1`

See also PDF

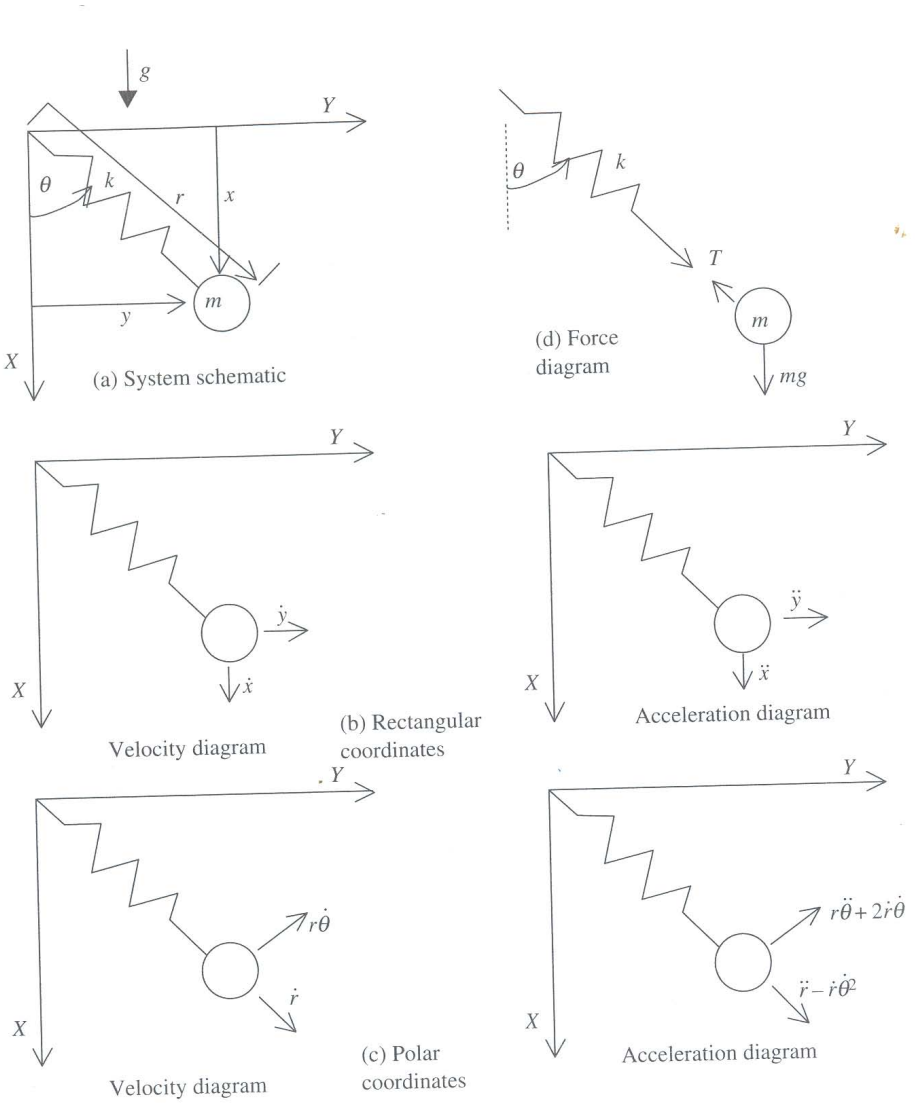
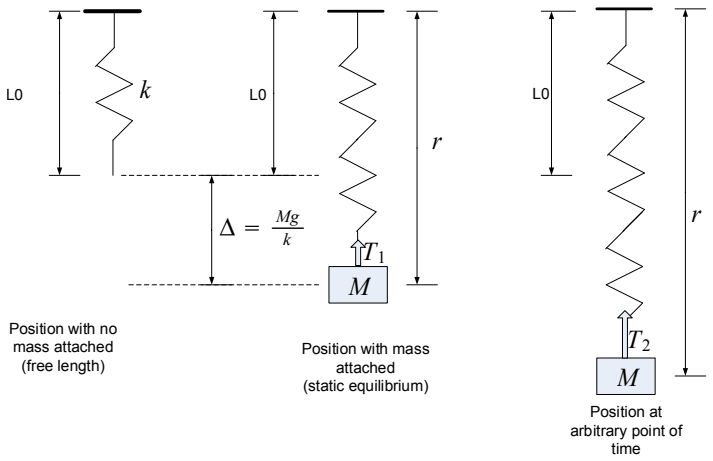


Figure 2.1 Spring–pendulum example system.

Notice that the *r* generalized coordinate is measured from the corner where the spring is attached to and not from the static equilibrium position. This diagram below helps to illustrate this more.



$$T_1 = k(r - l_0) = k\Delta$$
$$T_2 = k(r - l_0)$$

Now the equations of motion are found and then they are converted to state form description and then ODE solver was used to integrate them.

4.1.3 Mathematical model

This system is 2 degree of freedom system, since two generalized coordinates are required to uniquely locate the mass m at any point of time. Hence, two EQM's (equation of motions) are required.

Applying $F = ma$ in the radial spring direction results in the following EQM

$$-T + mg\cos\theta = m(r'' - r\,(\theta')^2)$$

Rearranging, and noting that $T = k\,(r - l_0)$ we obtain

$$r'' = r\,(\theta')^2 + g\cos\theta - \frac{k}{m}\,(r - l_0) \tag{1}$$

Applying $F = ma$ in the perpendicular direction to the spring results in the following EQM

$$\begin{aligned} -mg\sin\theta &= m\,(r\theta'' + 2r'\theta') \\ \theta'' &= -\frac{g}{r}\sin\theta - \frac{2r'\theta'}{r} \end{aligned} \tag{2}$$

To use the ODE solver, the above second order ODE's, Eqs., (1) and (2), need to be converted to first order ODE. This is done by introducing state variables as follows

$$\left.\begin{array}{l} x_1 = r \\ x_2 = r' \\ x_3 = \theta \\ x_4 = \theta' \end{array}\right\} \overset{\frac{d}{dt}}{\rightarrow} \left\{\begin{array}{l} x_1' = x_2 \\ x_2' = x_1 x_4^2 + g\cos x_3 - \frac{k}{m}\,(x_1 - l_0) \\ x_3' = x_4 \\ x_4' = -\frac{g}{r}\sin x_3 - \frac{2x_2 x_4}{x_1} \end{array}\right\}$$

Therefore, the state space form is

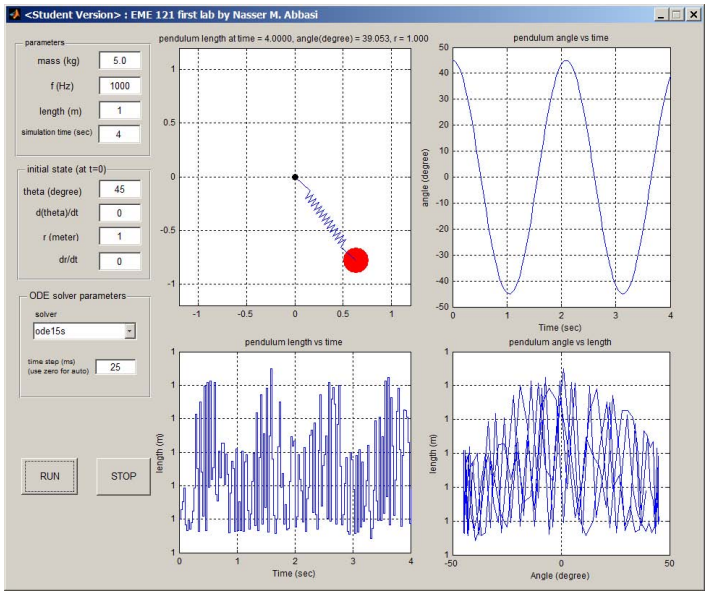
$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 x_4^2 + g\cos x_3 - \frac{k}{m}\,(x_1 - l_0) \\ x_4 \\ \frac{g}{r}\sin x_3 - \frac{2x_2 x_4}{x_1} \end{pmatrix} \tag{3}$$

The above system is solved using ODE solver and the result are shown in the following section.

4.1.4 Result of simulation

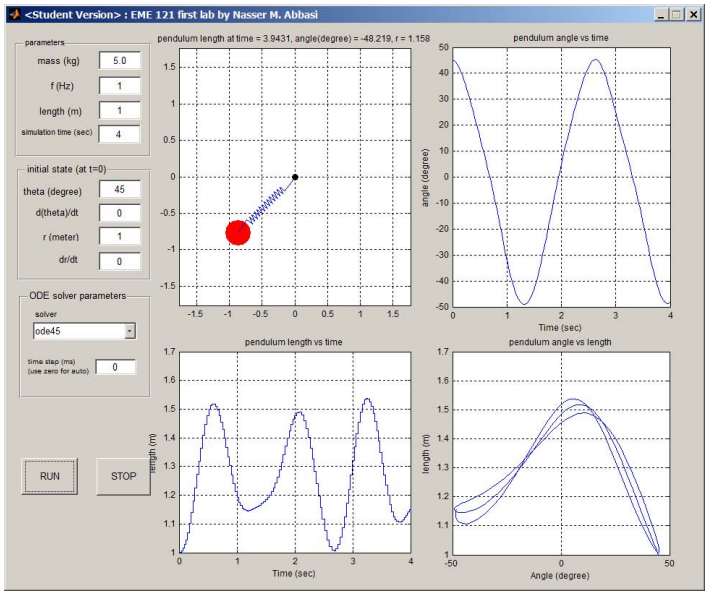
Using the following parameters to generate figure (3.2) and figure 3.3 for the very stiff spring

m	f	l_0	$r\,(0)$	$r'\,(0)$	$\theta\,(0)$	$\theta'\,(0)$
$5kg$	$1000\ Hz$	$1\ m$	$1\ m$	0	45^0	0



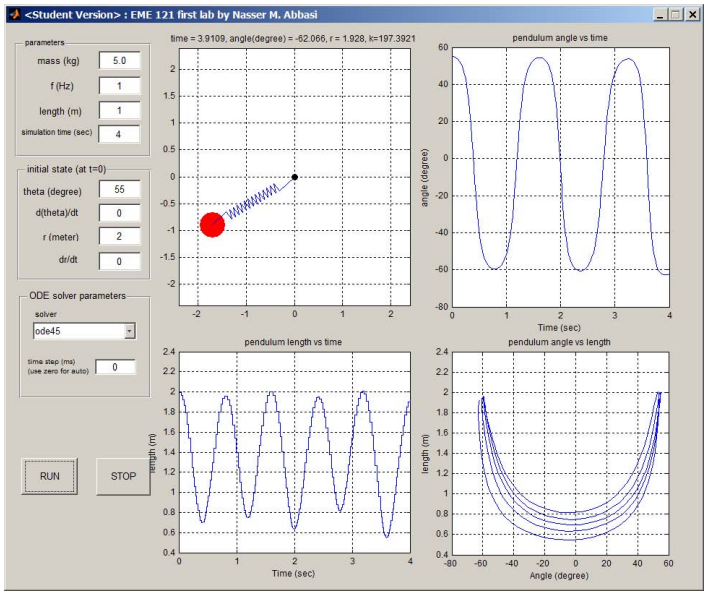
Using the following parameters to generate figure (3.4)

m	f	l_0	$r(0)$	$r'(0)$	$\theta(0)$	$\theta'(0)$
$5kg$	$1\ Hz$	$1\ m$	$1\ m$	0	45^0	0



Using the following parameters to generate figure (3.5)

m	f	l_0	$r(0)$	$r'(0)$	$\theta(0)$	$\theta'(0)$
$5kg$	$1\ Hz$	$1\ m$	$2\ m$	0	55^0	0



4.1.5 Discussion of results

When the spring is made very stiff, then the change of the pendulum length can be seen from the figure 3.3 result to be very small. This is the same result as a normal pendulum will have. This is also verified by figure 3.2 which shows a simple harmonic motion as shown in figure 3.2 (pendulum angle vs. time).

When the spring stiffness is reduced ($f = 1Hz$), then we can see that the motion is no longer a simple harmonic motion as can be seen in figure (3.3 and 3.4). Changing the length of the initial pendulum also resulted is completely different profile of the motion.

4.1.6 Appendix, Surce code listing

```
function varargout = eme_121_lab1(varargin)
% EME_121_LAB1 M-file for eme_121_lab1.fig
%      EME_121_LAB1, by itself, creates a new
%      EME_121_LAB1 or raises the existing singleton*.
%
%      H = EME_121_LAB1 returns the handle to a new
%          EME_121_LAB1 or the handle to the existing singleton*.
%
%      EME_121_LAB1('CALLBACK',hObject,eventData,handles,...)
%        calls the local function named CALLBACK in EME_121_LAB1.M
%        with the given input arguments.
%
%      EME_121_LAB1('Property','Value',...) creates a new
%           EME_121_LAB1 or raises the existing singleton*.
%           Starting from the left, property value pairs are applied
%           to the GUI before eme_121_lab1_OpeningFcn gets called.  An
%           unrecognized property name or invalid value makes
%           property application stop.  All inputs are passed
%           to eme_121_lab1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.
%       Choose "GUI allows only one instance to run (singleton)".
%
```

```
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help eme_121_lab1

% Last Modified by GUIDE v2.5 10-Apr-2011 23:32:56

% By Nasser M. Abbasi

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @eme_121_lab1_OpeningFcn, ...
    'gui_OutputFcn',  @eme_121_lab1_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before eme_121_lab1 is made visible.
function eme_121_lab1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to eme_121_lab1 (see VARARGIN)

% Choose default command line output for eme_121_lab1
handles.output = hObject;

set(handles.figure1, 'UserData',[]);
set(handles.figure1,'Name','EME 121 first lab by Nasser M. Abbasi');

data.stop = false;
set(handles.figure1, 'UserData',data);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes eme_121_lab1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = eme_121_lab1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function mass_tag_Callback(hObject, eventdata, handles)
% hObject    handle to mass_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mass_tag as text
% str2double(get(hObject,'String')) returns contents of mass_tag as a double


% --- Executes during object creation, after setting all properties.
function mass_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mass_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function f_tag_Callback(hObject, eventdata, handles)
% hObject    handle to f_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f_tag as text
% str2double(get(hObject,'String')) returns contents of f_tag as a double


% --- Executes during object creation, after setting all properties.
function f_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
           get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function length_tag_Callback(hObject, eventdata, handles)
% hObject    handle to length_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of length_tag as text
% str2double(get(hObject,'String')) returns contents of length_tag as a double


% --- Executes during object creation, after setting all properties.
```

```
function length_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to length_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function angle_zero_tag_Callback(hObject, eventdata, handles)
% hObject    handle to angle_zero_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of angle_zero_tag as text
% str2double(get(hObject,'String')) returns contents of angle_zero_tag as a double


% --- Executes during object creation, after setting all properties.
function angle_zero_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angle_zero_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function angle_speed_zero_Callback(hObject, eventdata, handles)
% hObject    handle to angle_speed_zero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of angle_speed_zero as text
%        str2double(get(hObject,'String')) returns contents of angle_speed_zero as a double


% --- Executes during object creation, after setting all properties.
function angle_speed_zero_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angle_speed_zero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function r_zero_tag_Callback(hObject, eventdata, handles)
% hObject    handle to r_zero_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of r_zero_tag as text
%        str2double(get(hObject,'String')) returns contents of r_zero_tag as a double


% --- Executes during object creation, after setting all properties.
function r_zero_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to r_zero_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function r_speed_zero_Callback(hObject, eventdata, handles)
% hObject    handle to r_speed_zero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of r_speed_zero as text
%        str2double(get(hObject,'String')) returns contents of r_speed_zero as a double


% --- Executes during object creation, after setting all properties.
function r_speed_zero_CreateFcn(hObject, eventdata, handles)
% hObject    handle to r_speed_zero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in run_btn.
function run_btn_Callback(hObject, eventdata, handles)
% hObject    handle to run_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[data,status] = parse_input(handles);
if not(status)
    return;
end

data.handles = handles;
data.g = 9.8;

enable_all(handles,'off');
userData = get(handles.figure1, 'UserData');
userData.stop = false;
set(handles.figure1, 'UserData',userData);

[g_msg,g_status]=process_eme_121_lab1(data);
if not(g_status)
```

```
    uiwait(errordlg(sprintf('Processing terminated: %s',g_msg),...
        'Bad Input', 'modal'));
    uicontrol(handles.f_tag);
end
enable_all(handles,'on');

%----------------------------
function [data,status]= parse_input(handles)

[data.mass,status]=verify_valid_positive_numeric...
    (get(handles.mass_tag,'String'),handles.mass_tag,...
    'Mass must be positive number');
if not(status)
    return;
end

[data.f,status]=verify_valid_positive_numeric...
    (get(handles.f_tag,'String'),handles.f_tag,...
    'natural frequency must be positive number');
if not(status)
    return;
end


[data.L,status]=verify_valid_positive_numeric...
    (get(handles.length_tag,'String'),handles.length_tag,...
    'pendulum length must be positive number');
if not(status)
    return;
end

[data.angle_zero,status]=verify_valid_numeric...
    (get(handles.angle_zero_tag,'String'),handles.angle_zero_tag,...
    'Initial angle must be numerical value');
if not(status)
    return;
end

if abs(data.angle_zero)>180
    uiwait(errordlg('Initial angle must be between 0 and 180 degrees only',...
        'Bad Input', 'modal'));
    status = 0;
    uicontrol(handles.angle_zero);
    return
else
    data.angle_zero = data.angle_zero*pi/180;
end

[data.r_zero,status]=verify_valid_positive_numeric...
    (get(handles.r_zero_tag,'String'),handles.r_zero_tag,...
    'initial r must be  numerical value');
if not(status)
    return;
end


[data.r_speed_zero,status]=verify_valid_numeric...
    (get(handles.r_speed_zero,'String'),handles.r_speed_zero,...
    'initial r speed must be numerical value');
if not(status)
    return;
```

```
end


[data.time_step,status]=verify_valid_numeric...
    (get(handles.time_Step_tag,'String'),handles.time_Step_tag,...
    'time step must be zero or larger');
if not(status)
    return;
end

[data.angle_speed_zero,status]=verify_valid_numeric...
    (get(handles.angle_speed_zero,'String'),handles.angle_speed_zero,...
    'initial angle speed must be numerical value');
if not(status)
    return;
end

[data.max_t,status]=verify_valid_positive_numeric...
    (get(handles.max_t_tag,'String'),handles.max_t_tag,...
    'maximum simulation time must be positive number');
if not(status)
    return;
end

contents = cellstr(get(handles.ode_solver_tag,'String'));
data.solver = contents{get(handles.ode_solver_tag,'Value')};


data.k = data.mass*(2*pi*data.f)^2;

%--------------------------------
function enable_all(handles,to)
set(handles.mass_tag,'Enable',to);
set(handles.f_tag,'Enable',to);
set(handles.length_tag,'Enable',to);
set(handles.angle_zero_tag,'Enable',to);
set(handles.r_speed_zero,'Enable',to);
set(handles.angle_speed_zero,'Enable',to);
set(handles.r_zero_tag,'Enable',to);
set(handles.max_t_tag,'Enable',to);
set(handles.ode_solver_tag,'Enable',to);
set(handles.time_Step_tag,'Enable',to);

function max_t_tag_Callback(hObject, eventdata, handles)
% hObject    handle to max_t_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of max_t_tag as text
%        str2double(get(hObject,'String')) returns contents of max_t_tag as a double


% --- Executes during object creation, after setting all properties.
function max_t_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to max_t_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in reset_tag.
function reset_tag_Callback(hObject, eventdata, handles)
% hObject    handle to reset_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


data = get(handles.figure1, 'UserData');
data.stop = true;
set(handles.figure1, 'UserData',data);
enable_all(handles,'on');


% --- Executes on selection change in ode_solver_tag.
function ode_solver_tag_Callback(hObject, eventdata, handles)
% hObject    handle to ode_solver_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ode_solver_tag contents as cell a
%        contents{get(hObject,'Value')} returns selected item from ode_solver_tag

% --- Executes during object creation, after setting all properties.
function ode_solver_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ode_solver_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function time_Step_tag_Callback(hObject, eventdata, handles)
% hObject    handle to time_Step_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of time_Step_tag as text
%        str2double(get(hObject,'String')) returns contents of time_Step_tag as a double


% --- Executes during object creation, after setting all properties.
function time_Step_tag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time_Step_tag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function  [g_msg,g_status]=process_eme_121_lab1(data)
% This function is called by the Matlab GUI to solve
% the equation of motion for Lab 1 problem, EME 121,
```

```
% spring 2011, UC Davis.
%
% INPUT: data, a record which contains the problem parameters
%
% The function will call ode45 solver and plot the solution
%
% by Nasser M. Abbasi
% 4/12/2011


% Reset axes for plotting
cla(data.handles.axes,'reset');
cla(data.handles.time_angle_axes,'reset');
cla(data.handles.time_r_axes,'reset');
cla(data.handles.angle_r_axes,'reset');

%Check if using auto step size of an explicit step size. see
%GUI for input choice.
if data.time_step==0
    time_span=[0 data.max_t];
else
    time_span=0:data.time_step/1000:data.max_t;
end

%set the y-axis for plotting
max_y = data.L;

%set the initial conditions for ode45
IC = [data.r_zero;data.r_speed_zero;data.angle_zero;data.angle_speed_zero];
options = odeset('OutputFcn',@output);

g_status = true;
g_msg='';

%call  the solver
[t,x] = feval(data.solver,@rhs,time_span,IC,options,data);


%plot the result
N=length(t);
picks=1:N;

set(data.handles.figure1, 'CurrentAxes',data.handles.time_angle_axes);
angle=x(:,3);
plot(t(picks),angle(picks)*180/pi,'-');
title('pendulum angle vs time');
xlabel('Time (sec)');
ylabel('angle (degree)');
grid;
drawnow;

set(data.handles.figure1, 'CurrentAxes',data.handles.time_r_axes);
r=x(:,1);
stairs(t(picks),r(picks),'-');
title('pendulum length vs time');
xlabel('Time (sec)'); ylabel('length (m)');
grid;
drawnow;

set(data.handles.figure1, 'CurrentAxes',data.handles.angle_r_axes);
```

```
plot(angle(picks)*180/pi,r(picks),'-');
title('pendulum angle vs length');
xlabel('Angle (degree)'); ylabel('length (m)');
grid;
drawnow;


%-----------------------------------
%
%-----------------------------------
    function dx=rhs(~,x,data)
        %the ode45 function
        temp = x(1)-data.L;

        if abs(temp<2*eps)
            term = 0;
        else
            term = data.k/data.mass*temp;
        end

        dx=[x(2);
            x(1)*x(4)^2+data.g*cos(x(3))-term;
            x(4);
            -2*x(2)*x(4)/x(1)- data.g/x(1)*sin(x(3))
            ];
        if x(1)>max_y
            max_y = x(1)+0.2*x(1);
        end
    end

%-----------------------------------
%
%-----------------------------------
    function status= output(t,x,~,data)
        %called by ode45 after each step. Plot the current
        %pendulum position for simulation

        userData = get(data.handles.figure1, 'UserData');
        if userData.stop
            status=true;
            g_status =true;
        else
            status = false;
            if not(isempty(t))
                plot_solution(x(1,end),x(3,end),t(1),data);
                %pause(0.1);
                if x(1,end)>10*data.L  %spring stretched too far
                    status = true;
                    g_status = false;
                    g_msg ='spring over stretched';
                end
            end
        end
    end

%-----------------------------------
%
%-----------------------------------
    function plot_solution(r,theta,t,data)
        %plot the final solution
        set(0,'CurrentFigure',data.handles.figure1);
```

```
        set(data.handles.figure1, 'CurrentAxes',data.handles.axes);

        [x_local,y] = pol2cart(theta-pi/2,r);
        plot(x_local,y,'o','MarkerSize',20,'MarkerEdgeColor','r',...
            'MarkerFaceColor','r','LineWidth',2);
        [x_local,y] = nma_spring.make(r,theta-(pi/2),30);
        hold on;
        line(x_local,y);
        plot(0,0,'o','MarkerSize',5,'MarkerEdgeColor','k',...
            'MarkerFaceColor','k','LineWidth',1);
        title(sprintf(...
            'time = %3.4f, angle(degree) = %3.3f, r = %3.3f, k=%3.4f',...
            t,theta*180/pi,r,data.k));

        xlim([-max_y max_y]);
        ylim([-max_y max_y]);
        %refreshdata
        grid;
        drawnow;
        hold off;
    end

end
classdef nma_spring
    %
    %static class to make spring for plotting animations
    %by Nasser M. Abbasi


    properties
    end

    methods(Static)

        %----------------------------
        %
        %----------------------------
        function [x,y] = make(r,theta,N)
            %r: total length of spring
            %theta: in radians, anticlock wise is positive,
            %       theta zero is position x-axis
            %N: number of twists in spring
            %
            %OUTPUT:
            % x,y coordinates of line to use to plot spring
            len = (4/6)*r;
            p = zeros(N,2);
            delr = len/N;

            r0     = (1/6)*r;
            p(2,1) = r0;
            p(2,2) = theta;

            for n=3:N-2
                p(n,1)=r0+delr*n;
                z=atan(2*delr/p(n,1));
                p(n,2)=theta+(-1)^n*z;
            end
            p(end-1,1)=(5/6)*r; p(end-1,2)=theta;
            p(end,1)=r; p(end,2)=theta;
```

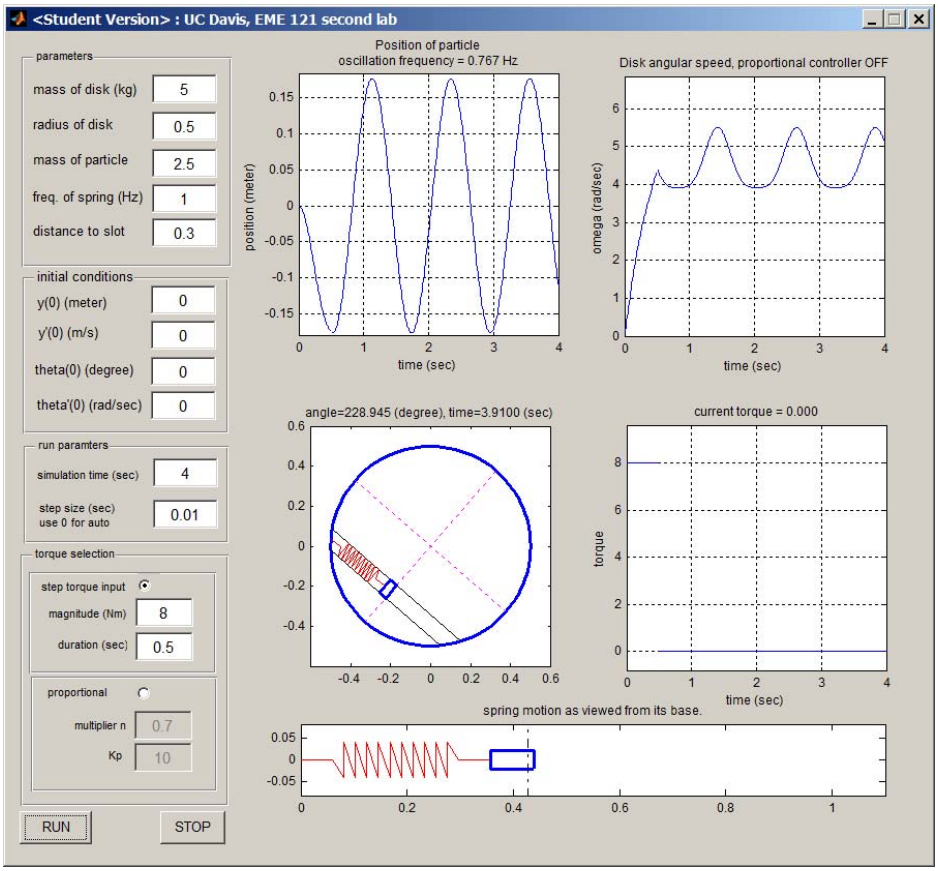
```
    [x,y] = pol2cart(p(:,2),p(:,1));
end

%----------------------------------------
function [x,y] = makeBox(w,h)
    x = zeros(5,1);
    y = zeros(5,1);
    x(1)= w/2; y(1)=0;
    x(2)= w/2; y(2)=h/2;
    x(3)= -w/2; y(3)=h/2;
    x(4)=-w/2;  y(4)=0;
    x(5)=x(1); y(5)=y(1);

end
%---------------------------
%
%---------------------------
function [x,y] = makeV2(len,N,w)
    %len: total length of spring
    %N: number of twists in spring
    %
    %OUTPUT:
    % x,y coordinates of line to use to plot spring

    nCoordinates = 2*N+4;
    p = zeros(nCoordinates,2);
    h = (1/6)*len;
    del=(len-2*h)/(N+1);
    done = false;
    n = 1;
    while not(done)

        if n==1
           p(1,1)=0;
           p(1,2)=0;
           n = n + 1;
        elseif n==2
           p(2,1)=0;
           p(2,2)=h;
           n = n + 1;
        elseif n<nCoordinates-2
           p(n,1)=w;
           p(n,2)=p(n-1,2)+del;
           n = n  + 1;
           p(n,1)=-w;
           p(n,2)=p(n-1,2);
           n = n + 1;
        elseif n==nCoordinates-1
           p(n,1) = 0;
           p(n,2)=p(n-1,2)+del;
           n = n + 1;
        else
           p(n,1) = 0;
           p(n,2)=p(n-1,2)+h;
           done = true;
        end
    end
    x=p(:,1);
    y=p(:,2);

end
```

```
%------------------------
%
%------------------------
function test(theta)

    close all;

    theta = theta*pi/180;
    figure;
    set(gcf,'Units','normalized');
    delr=0.3;
    for r=1:.1:3
        cla;
        [x,y] = nma_spring.make(r,theta,30);
        line(x,y);
        hold on;

        %baseX = [2*delr*cos((pi/2)-theta) -2*delr*sin(theta)];
        %baseY = [-2*delr*sin((pi/2)-theta) 2*delr*cos(theta)];

        baseX = [-2*delr*sin(abs(theta)) 2*delr*sin(abs(theta))];
        baseY = [-2*delr*cos(theta)      2*delr*cos(theta)];
        line(baseX,baseY);

        triX=[baseX(1) ...
            baseX(1) ...
            30*delr*cos(theta)-abs(baseX(1))...
            baseX(1)];

        triY=[baseY(1) ...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
            baseY(1)];

        line(triX,triY);

        L=sqrt(x(end)^2+y(end)^2);
        xx0=L*cos(theta);
        yy0=-L*sin(abs(theta));
        massX=[xx0 ...
            xx0+2*delr*sin(abs(theta)) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-...
                  4*delr*sin(abs(theta))...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-...
                  4*delr*sin(abs(theta))-4*delr*cos(theta)...
            xx0];

        massY=[ yy0 ...
            yy0+2*delr*cos(theta) ...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta)) ...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-...
                     4*delr*cos(theta)...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-...
                      4*delr*cos(theta)+4*delr*sin(abs(theta))...
            yy0];

        line(massX,massY);
```

```
        xlim([-3,10]); ylim([-10,3]);

        set(gca,'DataAspectRatioMode','manual',...
                  'DataAspectRatio',[1 1 1]);
        drawnow;
        pause(.1);
    end

end

%-------------------------------
%
%-------------------------------
function testV2(len)

    close all;

    theta = 0;
    figure;
    set(gcf,'Units','normalized');
    delr=0.3;
    for r=1:len/3:len
        cla;
        [x,y] = nma_spring.makeV2(r,10,1);
        line(x,y);
        hold on;

        %baseX = [2*delr*cos((pi/2)-theta) -2*delr*sin(theta)];
        %baseY = [-2*delr*sin((pi/2)-theta) 2*delr*cos(theta)];

        baseX = [-2*delr*sin(abs(theta)) 2*delr*sin(abs(theta))];
        baseY = [-2*delr*cos(theta)      2*delr*cos(theta)];
        line(baseX,baseY);

        triX=[baseX(1) ...
            baseX(1) ...
            30*delr*cos(theta)-abs(baseX(1))...
            baseX(1)];

        triY=[baseY(1) ...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
            baseY(1)];

        line(triX,triY);

        L=sqrt(x(end)^2+y(end)^2);
        xx0=L*cos(theta);
        yy0=-L*sin(abs(theta));
        massX=[xx0 ...
            xx0+2*delr*sin(abs(theta)) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-...
                     4*delr*sin(abs(theta))...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-...
                     4*delr*sin(abs(theta))-4*delr*cos(theta)...
            xx0];

        massY=[ yy0 ...
            yy0+2*delr*cos(theta) ...
```

```
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta)) ...
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-...
                           4*delr*cos(theta)...
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-...
                           4*delr*cos(theta)+4*delr*sin(abs(theta))...
                    yy0];

                line(massX,massY);

                xlim([-3,10]); ylim([-10,3]);

                set(gca,'DataAspectRatioMode','manual',...
                       'DataAspectRatio',[1 1 1]);
                drawnow;
                pause(.3);
            end

        end

    end
end
```

4.2 Lab 2, Simulate moving mass on spring inside slot on a moving table



This Lab assignment is to study motion of circular disk with mass on spring moving in a slot on the table.

4.2.1 Problem description

The simulation assignment is to reproduce Example 3.2 in your book for the pinned disk with a slot and mass particle. Turn in your computer file with the equations you simulated and plots that show that your simulation worked.

An addendum to the problem is in this

Additions to Simulation 2, disk with slot and mass particle

In class it was shown that if the disk was operating at constant angular velocity then the mass particle would oscillate at a frequency that was dependent on the rotational speed of the disk. In fact, the oscillation frequency would decrease as the angular velocity increases.

Now that you have a model operating, it would be interesting to test the validity of the analysis done in class. To do this we need a controller that will keep the disk at constant rotational speed. Construct a proportional controller such that the input torque is proportional to the angular speed error,

$\tau_{in} = K_p(\omega_{des} - \omega)$

From the parameters in your text,

$\omega_n = 2\pi f_n$.

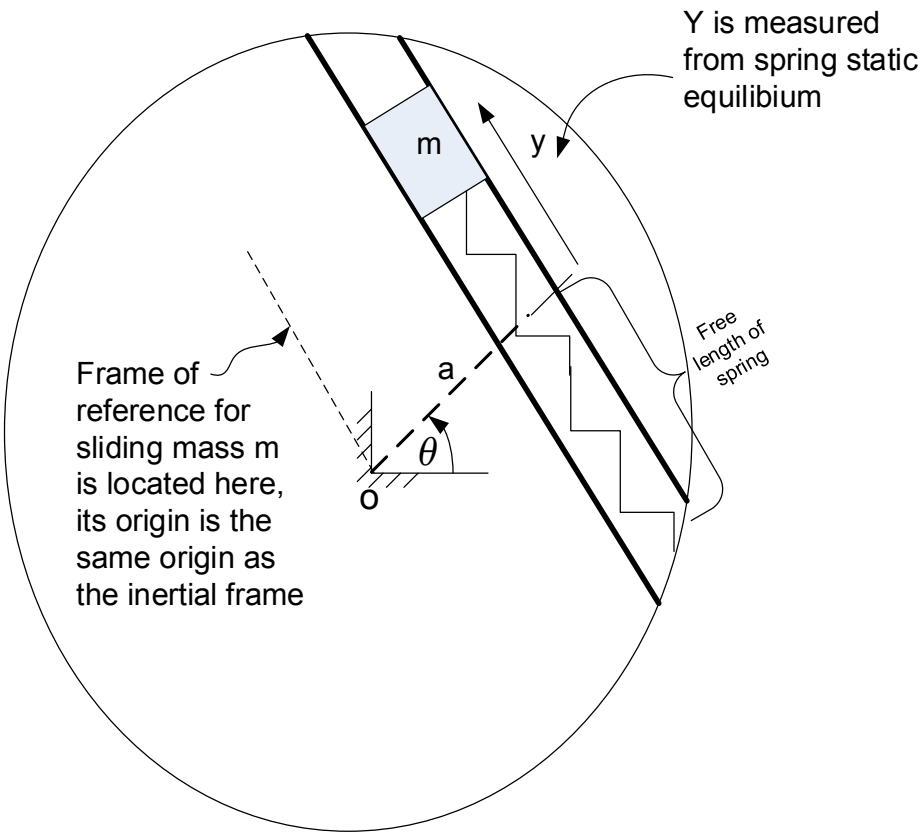
Try $\omega_{des} = n\omega_n \quad n = 0.5, 0.7, 0.9, 0.99$

I found that $K_p = 10$ works well.

Plot the motion of the particle for each of the test cases. You will find that the particle motion damps out. But you should still be able to determine if the oscillation frequency does decrease as the rotational speed increases. Comment on your observations. What happens if $\omega_{des} > \omega_n$?

4.2.2 Physical model

The following diagram shows the physical model of the system. It is a horizontal table, with a slot in it, where a mass *m* attached to spring that moves with no friction inside this slot.



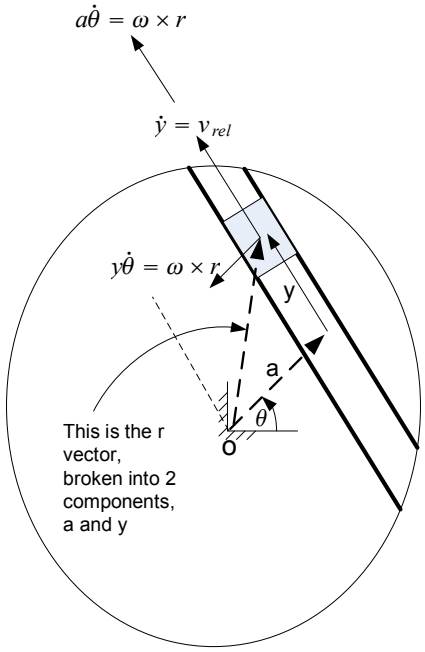
In the above, the frame of reference for the moving mass m has its origin coincide with the origin of the inertial frame. The vector a is always perpendicular to the slot, and it is assumed the mass m when the spring is relaxed will be located at the end of the vector a and hence y is measured related to that location.

4.2.3 Mathematical model

The system has 2 degrees of freedom θ and y. The 2 equations of motion will be derived using both the $F = ma$ and using Lagrangian method.

Deriving equations of motion using $F = ma$

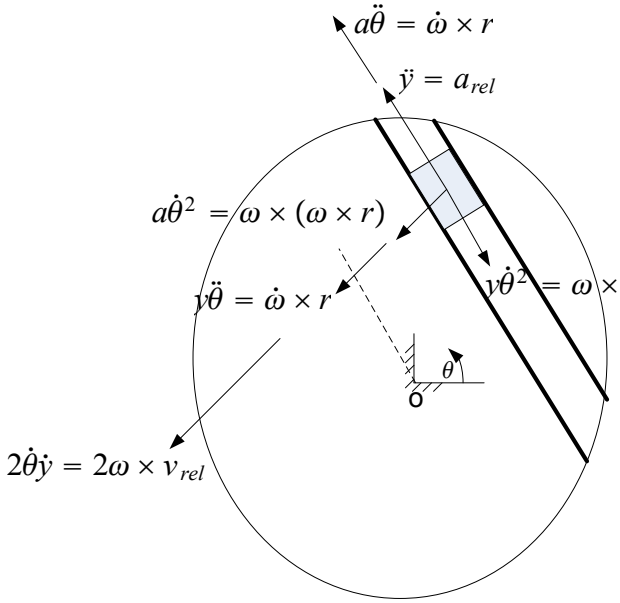
The following is the velocity, acceleration and force diagrams



Velocity diagram

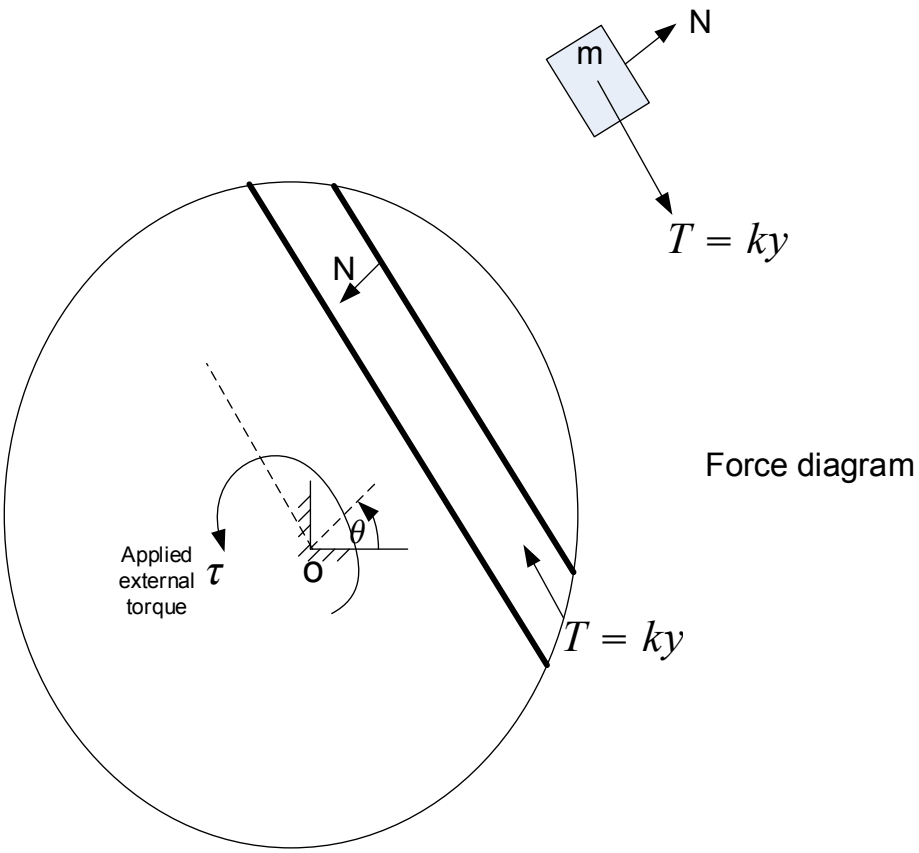
$$v_p = v_o + \omega \times r + v_{rel}$$

ω vector points outside the page



accerlaration diagram

$$a_p = a_o + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_{rel} + a_{rel}$$



Force diagram

Now that we have the acceleration and the force diagrams, we can write the equations of motion using $F = ma$.

There are 3 unknowns in the problem θ'', y'' and N, the reaction force on the side of the slot wall. Hence we need to generate 3 equations.

Apply $F = ma$ in the direction parallel to the a vector we obtain

$$\sum F = -m\left(a(\theta')^2 + y\theta'' + 2\theta' y'\right)$$
$$N = -m\left(a(\theta')^2 + y\theta'' + 2\theta' y'\right) \quad (1)$$

Applying $F = ma$ in the direction parallel to the y vector gives

$$\sum F = m\left(a\theta'' + y'' - y(\theta')^2\right)$$
$$-ky = m\left(a\theta'' + y'' - y(\theta')^2\right)$$
$$y'' = -\frac{k}{m}y - a\theta'' + y(\theta')^2 \quad (2)$$

Applying moment equation for the disk results in

$$\tau + Ny + (ky)\,a = I_g\theta'' \quad (3)$$

Substituting Eq. (1 into Eq. (3) results in

$$\begin{aligned}\tau - \left[m\left(a\left(\theta'\right)^2 + y\theta'' + 2\theta' y'\right)\right] y + (ky)\, a &= I_g\theta'' \\ \tau - mya\left(\theta'\right)^2 - my^2\theta'' - 2m\theta' yy' + kya &= I_g\theta'' \\ \theta''\left(my^2 + I_g\right) &= \tau - mya\left(\theta'\right)^2 - 2my\theta' y' + kya\end{aligned}$$

Therefore, the final EQM are

$$y'' = -\frac{k}{m}y - a\theta'' + y\left(\theta'\right)^2 \tag{4}$$

$$\theta'' = \frac{\tau - mya\left(\theta'\right)^2 - 2my\theta' y' + kya}{my^2 + I_g} \tag{5}$$

Decoupling the EQM and the state space formulation

Substituting Eq. (5) into Eq. (4) gives

$$y'' = -\frac{k}{m}y - a\frac{\tau - mya\left(\theta'\right)^2 - 2my\theta' y' + kya}{my^2 + I_g} + y\left(\theta'\right)^2$$

Using the decoupled ODE above along with Eq. (5) and introducing 4 state variables $x_1, x_2, x_3\, x_4$ we obtain

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \theta \\ y \\ \theta' \\ y' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} \theta' \\ y' \\ \theta'' \\ y'' \end{pmatrix}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau}{my^2+I_g} - \frac{mya(\theta')^2}{my^2+I_g} - \frac{2my\theta' y'}{my^2+I_g} + \frac{kya}{my^2+I_g} \\ -\frac{k}{m}y - a\frac{\tau - mya(\theta')^2 - 2my\theta' y' + kya}{my^2+I_g} + y\left(\theta'\right)^2 \end{pmatrix}$$

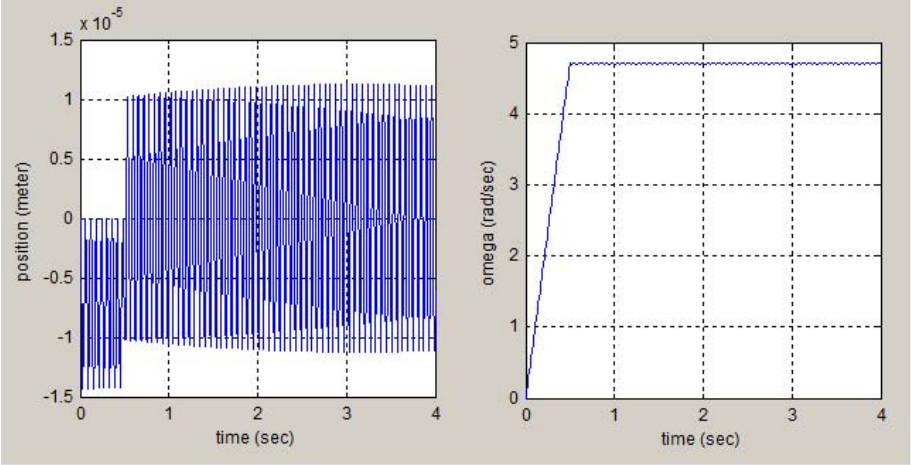
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau}{mx_2^2+I_g} - \frac{mx_2 ax_3^2}{mx_2^2+I_g} - \frac{2mx_2 x_3 x_4}{mx_2^2+I_g} + \frac{kx_2 a}{mx_2^2+I_g} \\ -\frac{k}{m}x_2 - a\left[\frac{\tau}{mx_2^2+I_g} - \frac{mx_2 ax_3^2}{mx_2^2+I_g} - \frac{2mx_2 x_3 x_4}{mx_2^2+I_g} + \frac{kx_2 a}{mx_2^2+I_g}\right] + x_2 x_3^2 \end{pmatrix}$$

Simulation results and discussion

First, the 4 figures shown in the textbook at page 100 and 101 are reproduced below using the simulation program written for this assignment.

Figure 3.7 and 3.8 reproduced

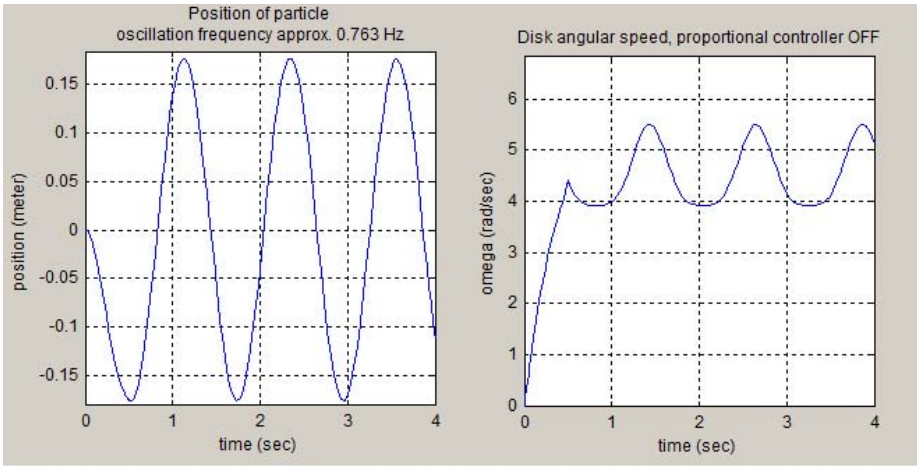
When the spring is made very stiff (f_n = 200 Hz) then the mass attached to the spring had almost no moment. Therefore, the system behaved as if the mass was just placed on the table, and no reactive force was generated on the side of the table due to spring extension. The generated solution agrees with this result as shown below.



All other simulation parameters used are as shown in table 3.2, page 99 in the textbook.

Figure 3.9 and 3.10 reproduced

Now the simulation parameters were all set to the same values in table 3.2, and the following is the result:



We can see that now, since the spring is extended and contracting, then there will be reactive force on the table at the point where the spring is attached to it. This force will change in value as the mass move

up and down the slot. Therefore this resulted in the disk angular speed changing as shown above. We notice now that the mass position produces oscillatory motion around the point of static equilibrium of the spring.

Results when proportional controller added

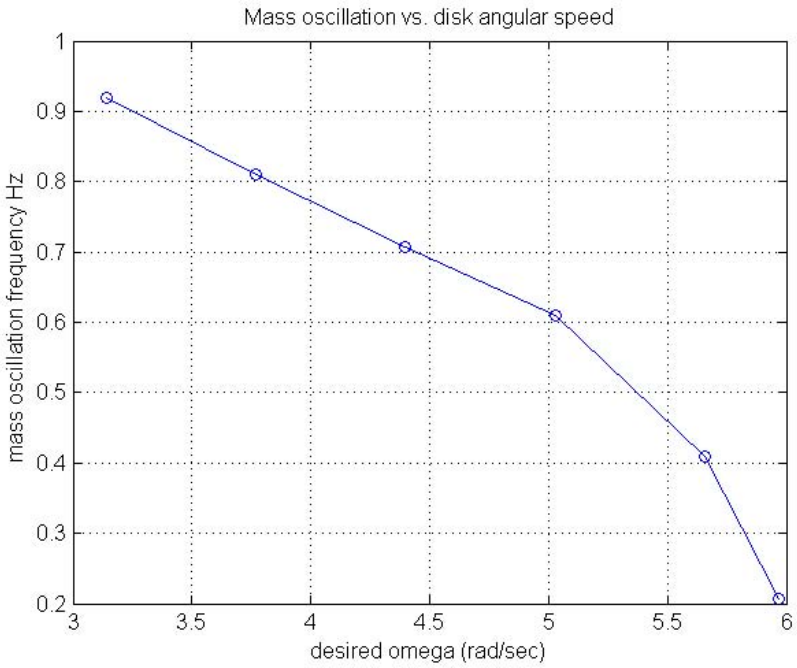
The input torque applied to the disk is now given by

$$\tau = K_p \left(\omega_{des} - \omega\right)$$

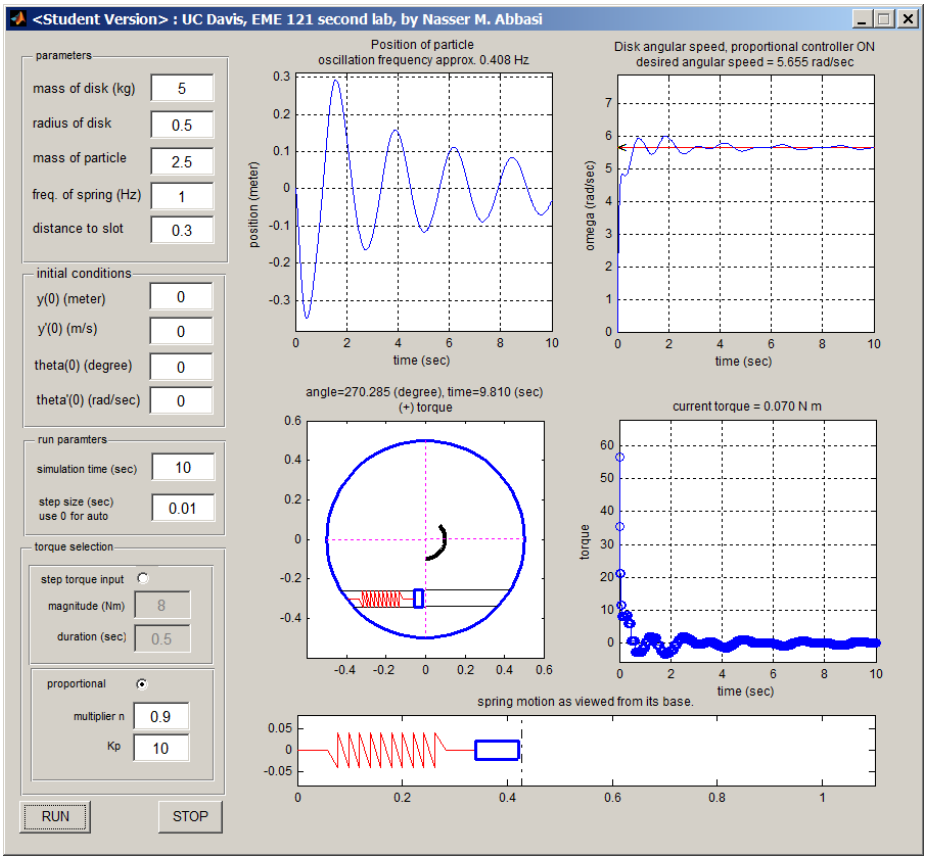
where ω_{des} is the angular speed of the disk that we would like the disk to be running at all the time, and ω is the actual angular speed of the disk. Different values of ω_{des} was tried. It was found that as the ω_{des} was made larger (in other words, as the disk angular speed was made larger and constant) then the oscillation of the mass attached to the spring became smaller. Hence there is an inverse relationship. Therefore, by measuring the oscillation frequency of the mass attached to the spring, one can determine the angular velocity of the disk. This is the idea behind tachometer. The following table shows the different ω_{des} used, and the resulting oscillation frequency of the mass. All other parameters used are as shown in table 3.2 in the textbook. K_p used was set to 10 in all the runs. Simulation time was set to 10 seconds.

n	$\omega_{des}\,(rad/\text{sec})$	mass oscillation (Hz)
0.5	3.142	0.92
0.6	3.77	0.81
0.7	4.398	0.708
0.8	5.027	0.609
0.9	5.655	0.41
0.95	5.969	0.306
0.99	6.22	mass fell off the edge

We notice that as the ω_{des} approached or exceeded the natural frequency of the spring, then the mass will oscillate such that the amplitude became larger than the distance it has available to it on the top of the disk, and the mass fell over, and the simulation was stopped then. The simulation program detects when the mass position is outside the disk and will terminate the simulation. This can be attributed to resonance, since at resonance the amplitude becomes very large. The following plot shows the above result to confirm the inverse relation between mass oscillation and the disk angular speed



The mass position amplitude damped out during oscillation in all the above runs was, but the oscillation was clear and was measured in the program by counting how many times the mass crosses its position of static equilibrium. The following plot shows the output from one typical run showing the damped motion of the mass.


<Student Version> : UC Davis, EME 121 second lab, by Nasser M. Abbasi
parameters
mass of disk (kg) 5
radius of disk 0.5
mass of particle 2.5
freq. of spring (Hz) 1
distance to slot 0.3
initial conditions
y(0) (meter) 0
y'(0) (m/s) 0
theta(0) (degree) 0
theta'(0) (rad/sec) 0
run paramters
simulation time (sec) 10
step size (sec) use 0 for auto 0.01
torque selection
step torque input
magnitude (Nm) 8
duration (sec) 0.5
proportional
multiplier n 0.9
Kp 10
RUN
STOP
Position of particle
oscillation frequency approx. 0.408 Hz
position (meter)
time (sec)
Disk angular speed, proportional controller ON
desired angular speed = 5.655 rad/sec
omega (rad/sec)
time (sec)
angle=270.285 (degree), time=9.810 (sec)
(+) torque
current torque = 0.070 N m
torque
time (sec)
spring motion as viewed from its base.

4.3 Lab 3, Simulation of crank-piston system

4.3.1 Problem description

Simulation 3 Slider-Crank Mechanism with a Piston

In your text, the slider-crank kinematic mechanism is discussed and equations of motion derived using Lagrange's equations. Shown in Figure 1 is this device. The flywheel has moment of inertia J_{fw}, crank radius R and the connecting rod is of length L. There is an input torque τ_{in} acting on the flywheel. Between the end of the connecting rod and the piston are a spring and damper k, b that represent the very high stiffness between the piston and connecting rod. These elements also help in obtaining an easily computable set of equations. The piston has mass m_p.

Your job is to:
1) Derive the equations of motion for this system using Lagrange's equations

2) Cast your equations into first order form suitable for simulation.

3) Simulate the system using the parameters and instructions in Table 1.

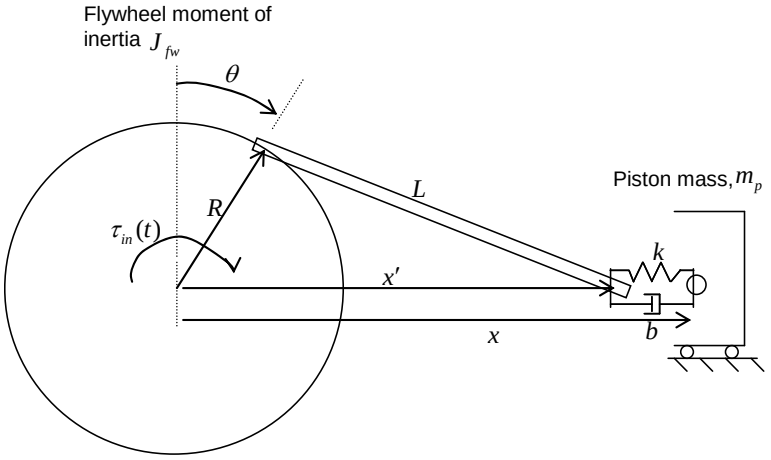


Figure 1 Schematic of the slider-crank system

Weight of flywheel=2.0 lb $\therefore\ m_{fw} = 2.0 / 2.2 Kg$
Volume of the cylinder(not shown)=500cc
Diameter of piston $D_p = 3.0in = 3.0(.0254)m$ $A_p = \frac{\pi}{4} D_p^2$
Stroke= $\frac{\text{piston volume}}{A_p}$
$R = \frac{\text{stroke}}{2}$
$\frac{R}{L} = 0.75$
$J_{fw} = m_{fw} \frac{R^2}{2}$
$m_p R^2 = 0.25 J_{fw}$
Define a frequency, f_n $\omega_n = 2\pi f_n$ and define a damping ratio, $\varsigma = 0.2$ then, $k = m_p \omega_n^2$ $b = 2\varsigma \omega_n m_p$
Maximum RPM=5000rpm

Table 1 Parameters for the slider-crank system

It is desired to accelerate the flywheel from 0 rpm to 5000 rpm in about 1 second. Experiment with the input torque τ_{in} to make this happen. When the angular speed reaches 5000 rpm, set the input torque to zero and observe what happens.

You will also need to experiment with f_n such that the deflection of the spring is not "too" big. It would seem reasonable if the deflection remained below a fraction of a millimeter.

Output representative plots and discuss your observations. Make sure and create an output that is the force on the piston and also plot the acceleration of the piston in g's. Turn in your program files and plots along with your written observations.

4.3.2 Animation

The GUI based simulation was written in Matlab 2011a. A number of animated GIF files were made. In the table below, clicking on any image will start a gif animation in your browser.

These are relatively large animation GIF files and might take 1-2 seconds to load depending on network bandwidth. To run them locally, they can be downloaded as well.

4.3.3 Part 1

The Lagrangian energy method was used to derive the equations of motion. The physical model used to represent the connection between the piston and the crank arm uses the method of Knarnopp-Margolis. This method uses a virtual spring and virtual damper for the connection between the piston and the crank arm instead of a rigid connection. This leads to a simpler mathematical model.

The equation of motion will now be derived based on this method.

Let $f(\theta)$ be the distance which is shown in the diagram above as x', The kinematic constraint is the following

$$f(\theta) = R\sin\theta + \sqrt{L^2 - R^2\cos^2\theta}$$

The extension in the spring is

$$\Delta = x - f(\theta)$$

Hence the spring force is

$$F_s = k\left[x - f\theta)\right] \tag{1}$$

The damping force is

$$\begin{aligned} F_d &= b\dot{x} \\ &= b\frac{d}{dt}\left(x - f(\theta)\right) \\ &= b\left(\dot{x} - \frac{df(\theta)}{d\theta}\dot{\theta}\right) \end{aligned} \tag{2}$$

The system is a 2 DOF, with generalized coordinates θ, x.

Let $I = \frac{MR^2}{2}$ be the moment of inertia of the disk (called J_{fw} in the above diagram) around an axis through the disk center, where M is the mass of the disk (called m_{fw} in the diagram). Let m be the mass of the cylinder (called m_p in the problem diagram).

The kinetic energy of the system is

$$T = \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\dot{x}^2$$

The potential energy is

$$\begin{aligned} V &= \frac{1}{2}k\Delta^2 \\ &= \frac{1}{2}k\left(x - f(\theta)\right)^2 \end{aligned}$$

Therefore the Lagrangian is

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}m\dot{x}^2 - \frac{1}{2}k\left(x - f(\theta)\right)^2 \end{aligned}$$

For θ we find

$$\begin{aligned} \frac{\partial L}{\partial\dot{\theta}} &= I\dot{\theta} \\ \frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}} &= I\ddot{\theta} \end{aligned}$$

and

$$\begin{aligned} \frac{dL}{d\theta} &= -k\left(x - f(\theta)\right)\left(-\frac{df(\theta)}{d\theta}\right) \\ &= k\left(x - f(\theta)\right)\frac{df(\theta)}{d\theta} \end{aligned}$$

Therefore the EQM for θ is

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}} - \frac{dL}{d\theta} = Q_\theta \tag{3}$$

To find the generalized force Q_θ, we make a virtual $\delta\theta$ displacement while keeping x fixed and then find the virtual work done. This results in

$$\begin{aligned}\delta W &= \tau\delta\theta + F_d\left(\delta f\left(\theta\right)\right)\\ &= \tau\delta\theta + F_d\frac{df\left(\theta\right)}{d\theta}\delta\theta\\ &= \left(\tau + F_d\frac{df\left(\theta\right)}{d\theta}\right)\delta\theta\end{aligned}$$

Hence from the above we see that

$$Q_\theta = \tau + F_d\frac{df\left(\theta\right)}{d\theta}$$

Eq. (3) becomes

$$I\ddot{\theta} - k\left(x - f\left(\theta\right)\right)\frac{df\left(\theta\right)}{d\theta} = \tau + F_d\frac{df\left(\theta\right)}{d\theta} \tag{4}$$

Now we evaluate $\frac{df(\theta)}{d\theta}$

$$\begin{aligned}f\left(\theta\right) &= R\sin\theta + \sqrt{L^2 - R^2\cos^2\theta}\\ \frac{df\left(\theta\right)}{d\theta} &= R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\end{aligned} \tag{5}$$

Substituting the above back into Eq. (2) gives

$$F_d = b\left(\dot{x} - \left(R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\dot{\theta}\right) \tag{6}$$

Substituting Eqs. (6,5) into Eq. (4) gives

$$\begin{aligned}I\ddot{\theta} - k\left(x - f\left(\theta\right)\right)\left(R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right) =\\ \tau + \left(b\left(\dot{x} - \left(R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\dot{\theta}\right)\right)\left(R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\end{aligned}$$

Simplifying the above gives the final EQM for θ

$$\begin{aligned}\ddot{\theta} = \frac{\tau}{I} + \frac{kR}{I}\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\left(\sqrt{L^2 - R^2\cos^2\theta} - R\sin\theta + x\right)\\ + \frac{bR}{I}\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\left(\dot{x} - R\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\dot{\theta}\right)\end{aligned} \tag{7}$$

Now to find the EQM for x

$$\begin{aligned}\frac{\partial L}{\partial\dot{x}} &= m\dot{x}\\ \frac{d}{dt}\frac{\partial L}{\partial\dot{x}} &= m\ddot{x}\end{aligned}$$

and

$$\frac{dL}{dx} = -k\left(x - f\left(\theta\right)\right)$$

Therefore the EQM for x is

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{x}} - \frac{dL}{dx} = Q_x \tag{8}$$

To find Q_x we make a virtual δx displacement while keeping θ fixed and find find the virtual work. Hence

$$\delta W = -F_d \delta x$$

Therefore $Q_x = F_d$ which was found in Eq. (2) above. Hence

$$Q_x = -b\left(\dot{x} - \frac{df(\theta)}{d\theta}\dot{\theta}\right)$$

And Eq. (8) becomes

$$m\ddot{x} + k\left(x - f(\theta)\right) = -b\left(\dot{x} - \frac{df(\theta)}{d\theta}\dot{\theta}\right)$$

Substituting the expressions for $f(\theta)$ and $\frac{df(\theta)}{d\theta}$ found earlier into the above results in

$$m\ddot{x} + k\left(x - \left(R\sin\theta + \sqrt{L^2 - R^2\cos^2\theta}\right)\right) = -b\left(\dot{x} - \left(R\cos\theta + \frac{R^2\cos\theta\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\dot{\theta}\right)$$

Therefore

$$\ddot{x} = \frac{k\sqrt{L^2 - R^2\cos^2\theta}}{m} + \frac{kR\sin\theta}{m} - \frac{kx}{m} - \frac{b\dot{x}}{m} + \frac{bR\dot{\theta}\cos\theta}{m} + \frac{b\dot{\theta}R^2\cos\theta\sin\theta}{m\sqrt{L^2 - R^2\cos^2\theta}} \tag{9}$$

4.3.4 Part 2

Now Eqs. (7) and (9) above will be cast into first order form.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \theta \\ x \\ \theta' \\ x' \end{pmatrix} \xrightarrow{\frac{d}{dt}} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \theta'' \\ x'' \end{pmatrix}$$

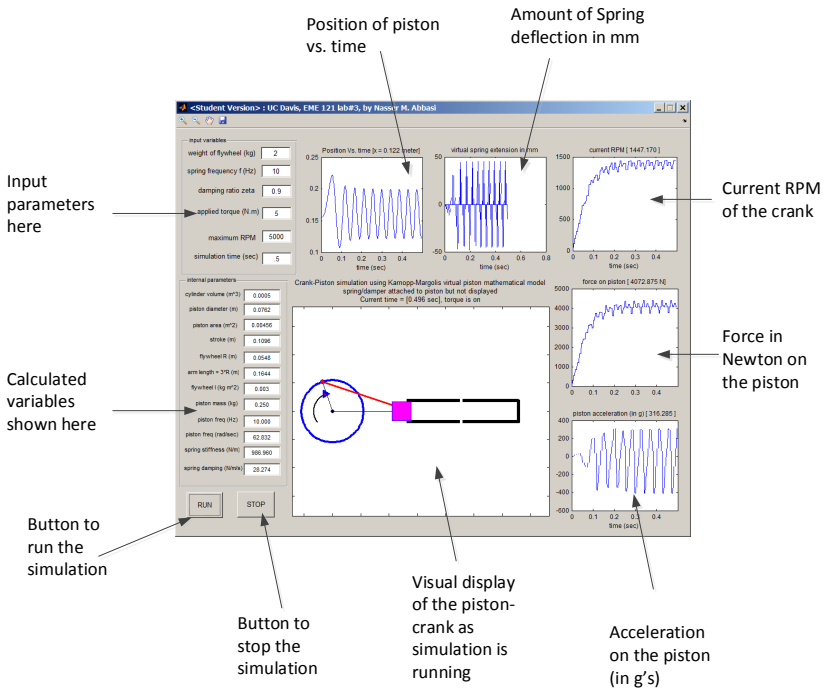
Hence

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau}{I} + \frac{kR}{I}\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\left(\sqrt{L^2 - R^2\cos^2\theta} - R\sin\theta + x\right) \\ \quad + \frac{bR}{I}\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\left(\dot{x} - R\cos\theta\left(1 + \frac{R\sin\theta}{\sqrt{L^2 - R^2\cos^2\theta}}\right)\dot{\theta}\right) \\ \frac{k\sqrt{L^2 - R^2\cos^2\theta}}{m} + \frac{kR\sin\theta}{m} - \frac{kx}{m} - \frac{b\dot{x}}{m} + \frac{bR\dot{\theta}\cos\theta}{m} + \frac{b\dot{\theta}R^2\cos\theta\sin\theta}{m\sqrt{L^2 - R^2\cos^2\theta}} \end{pmatrix}$$

$$= \begin{pmatrix} x_3 \\ x_4 \\ \frac{\tau}{I} + \frac{kR}{I}\cos x_1\left(1 + \frac{R\sin x_1}{\sqrt{L^2 - R^2\cos^2 x_1}}\right)\left(\sqrt{L^2 - R^2\cos^2 x_1} - R\sin x_1 + x_2\right) \\ \quad + \frac{bR}{I}\cos x_1\left(1 + \frac{R\sin x_1}{\sqrt{L^2 - R^2\cos^2 x_1}}\right)\left(x_4 - R\cos x_1\left(1 + \frac{R\sin x_1}{\sqrt{L^2 - R^2\cos^2 x_1}}\right)x_3\right) \\ \frac{k\sqrt{L^2 - R^2\cos^2 x_1}}{m} + \frac{kR\sin x_1}{m} - \frac{kx_2}{m} - \frac{bx_4}{m} + \frac{bRx_3\cos x_1}{m} + \frac{bx_3R^2\cos x_1\sin x_1}{m\sqrt{L^2 - R^2\cos^2 x_1}} \end{pmatrix}$$

4.3.5 Part 3

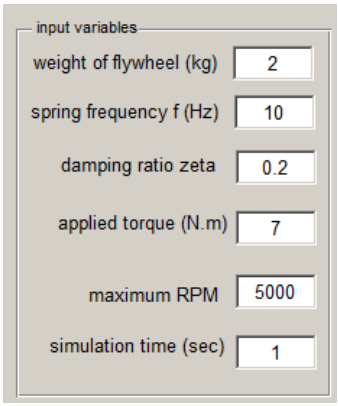
The system was simulated on the computer. The first order equations above were solved using ode numerical solver. A GUI program was written to make it easier to do the simulation and observe the results. The following describes the simulation program user interface



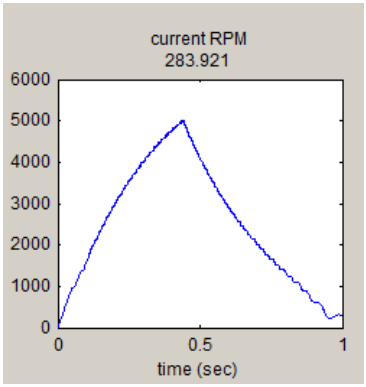
Crank to 5000 RPM

We are asked to find the torque needed to reach 5000 RPM in one second. By trial and error, a torque of $\tau = 7\ NM$ was found to meet this condition, with $f = 10Hz$, and damping ratio $\xi = 0.2$, and using the same other parameters shown in the problem table. Using this torque, the 5000 RPM was reached at about 0.5 seconds. The force on the piston started to become lower after the torque was removed, as well as the piston acceleration.

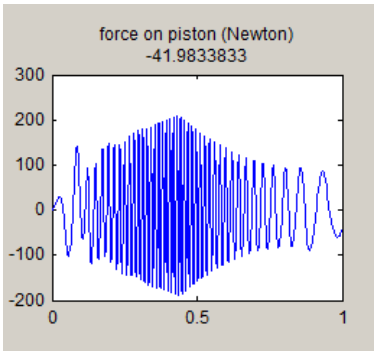
The following diagram shows the final plots at the end of the one second run.



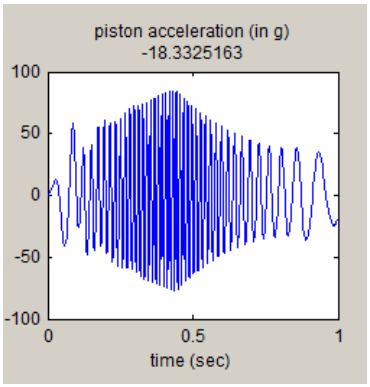
Parameters used



RPM plot. RPM 5000 was reached at 0.5 second using torque = 7 NM



Force on piston, showing force going down after torque is removed

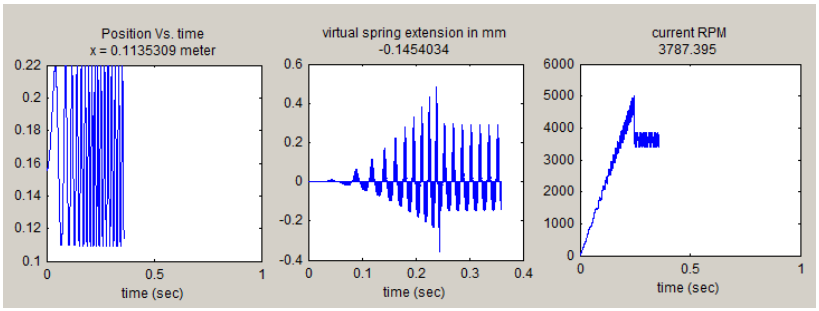


Acceleration of piston, showing reduction in acceleration after torque is removed

Simulation to keep spring deflection below fraction of mm

We are also asked to find the parameters which will cause the spring deflection to remain below fraction of a millimeter. It was found that the spring k must be made very large to simulate the effect of a rigid connection between the piston and the crank arm in order to keep the deflection very small. But when this was done, the simulation started to take very long time. Simulating 1 second on my computer would more than one hour.

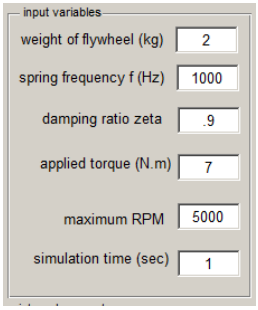
To reduce vibration, the damping coefficient ξ was made larger than the 0.2 value in the table. But even when making ξ close to one, the simulation still took very long time to reach one second due to the large number of steps taken by the numerical solver. The spring stiffness was increased all the way to 10^7 N/m, and the force on the piston reached $1.5MPa$ at one point. The value of f (spring frequency) to achieve this high spring stiffness was 1000 Hz. The damping ratio ξ was set to 0.9. The following is a plot of the simulation using the above parameters, showing that the deflection remained below half millimeter.



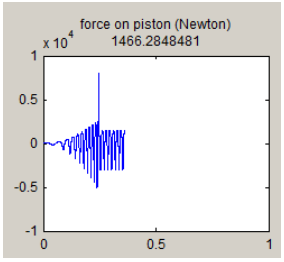
Position of piston Vs. time

Vibration in spring, kept below 1 mm during the whole simulation

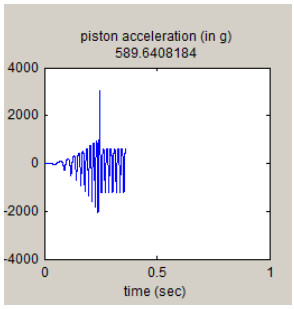
Showing the RPM cutoff at around .25 seconds



Parameters used



Force on piston



Acceleration of piston (in units of g)

Force and acceleration on piston

The acceleration on the piston was found numerically at each step by dividing the difference of the current speed of piston from the last time step speed, and dividing that over the difference in time as follows

$$a_x = \frac{\dot{x}(t_n) - \dot{x}(t_{n-1})}{t_n - t_{n-1}}$$

The force on the piston was found at each step as follows

$$F = -(F_s + F_d)$$

Where F_s is the force in spring and F_d is the damping force. Expression for these forces were derived above.

The force on the piston depended on the torque used and the amount of spring stiffness and damping used. For example, in the first test case above when the spring was made very stiff, the largest force on the piston reached was $1.5 \times 10^6 N$ and an acceleration of almost $1000g$ was seen. After the torque was removed, the acceleration gradually dropped to $500g$. The longer the simulation was run, these values became smaller. The simulation program updates both the acceleration and the force on the piston as the simulation is running.

General observation

The Knarnopp-Margolis model of the crank/piston connection was used to simulate the motion. This model is simpler mathematically than the rigid connection model. But it was hard to find the right

parameters to approximate a rigid connection since making the spring stiffness large, causes the simulation to become very slow. When the spring stiffness was made very large to better approximate a rigid connection, and when the damping was made larger, the simulation would take about 1 hour for 1 second simulation time. Even when using a stiff ode solver such as ode15s the simulation was taking too long. Using a compiled language would eliminate this problem as it will be much faster, but this simulation was done using Matlab

The main advantage of the Knarnopp-Margolis model is that the implementation was simpler since the mathematical model was simpler. But more trial and error are needed to find optimal values for all the parameters of the problem which will cause the simulation time to become more reasonable, while approximating the rigid connection.

4.3.6 Appendix, source code

```
function varargout = nma_lab3_eme_121_fig(varargin)
% NMA_LAB3_EME_121_FIG MATLAB code for nma_lab3_eme_121_fig.fig
%      NMA_LAB3_EME_121_FIG, by itself, creates a new NMA_LAB3_EME_121_FIG or raises the exist
%      singleton*.
%
%      H = NMA_LAB3_EME_121_FIG returns the handle to a new NMA_LAB3_EME_121_FIG or the handle
%      the existing singleton*.
%
%      NMA_LAB3_EME_121_FIG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in NMA_LAB3_EME_121_FIG.M with the given input arguments.
%
%      NMA_LAB3_EME_121_FIG('Property','Value',...) creates a new NMA_LAB3_EME_121_FIG or rais
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before nma_lab3_eme_121_fig_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to nma_lab3_eme_121_fig_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% this is the GUI main line for EME 121 lab 3
% by Nasser M. Abbasi
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help nma_lab3_eme_121_fig

% Last Modified by GUIDE v2.5 07-May-2011 17:21:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @nma_lab3_eme_121_fig_OpeningFcn, ...
                   'gui_OutputFcn',  @nma_lab3_eme_121_fig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before nma_lab3_eme_121_fig is made visible.
function nma_lab3_eme_121_fig_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to nma_lab3_eme_121_fig (see VARARGIN)

% Choose default command line output for nma_lab3_eme_121_fig
handles.output = hObject;

set(handles.figure1, 'UserData',[]);
set(handles.figure1,'Name','UC Davis, EME 121 lab#3, by Nasser M. Abbasi');

userData.stop = false;
set(handles.figure1, 'UserData',userData);


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes nma_lab3_eme_121_fig wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = nma_lab3_eme_121_fig_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function mfwTag_Callback(hObject, eventdata, handles)
% hObject    handle to mfwTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mfwTag as text
%        str2double(get(hObject,'String')) returns contents of mfwTag as a double


% --- Executes during object creation, after setting all properties.
function mfwTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mfwTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function fTag_Callback(hObject, eventdata, handles)
% hObject    handle to fTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fTag as text
%        str2double(get(hObject,'String')) returns contents of fTag as a double


% --- Executes during object creation, after setting all properties.
function fTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function zetaTag_Callback(hObject, eventdata, handles)
% hObject    handle to zetaTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of zetaTag as text
%        str2double(get(hObject,'String')) returns contents of zetaTag as a double


% --- Executes during object creation, after setting all properties.
function zetaTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zetaTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function torqueTag_Callback(hObject, eventdata, handles)
% hObject    handle to torqueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of torqueTag as text
%        str2double(get(hObject,'String')) returns contents of torqueTag as a double


% --- Executes during object creation, after setting all properties.
function torqueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to torqueTag (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function rpmTag_Callback(hObject, eventdata, handles)
% hObject    handle to rpmTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rpmTag as text
%        str2double(get(hObject,'String')) returns contents of rpmTag as a double


% --- Executes during object creation, after setting all properties.
function rpmTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rpmTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in runBtn.
function runBtn_Callback(hObject, eventdata, handles)
% hObject    handle to runBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

userData = get(handles.figure1, 'UserData');
userData.stop = false;
set(handles.figure1, 'UserData',userData);

data.angleInitial=(pi/2)/180;
data.angleSpeedInitial=0;

data.M=str2num(get(handles.mfwTag,'String'));
data.volumeOfCylinder=500*0.000001 ; %convery cc to m^3
data.Dp=3*.0254;
data.Ap=(pi/4)*data.Dp^2;
data.stroke=data.volumeOfCylinder/data.Ap;
data.R=data.stroke/2;
data.L=3*data.R;
data.I=data.M*data.R^2/2;
set(handles.Itag,'String',sprintf('%3.3f',data.I));

data.m=0.25*data.I/data.R^2;
set(handles.pistonMassTag,'String',sprintf('%3.3f',data.m));

data.f=str2num(get(handles.fTag,'String'));
set(handles.pistonFreqTag,'String',sprintf('%3.3f',data.f));
```

```
data.omega=2*pi*data.f;
set(handles.pistonWnTag,'String',sprintf('%3.3f',data.omega));


data.dampingRatio=str2num(get(handles.zetaTag,'String'));


data.k=data.m*data.omega^2;
set(handles.kTag,'String',sprintf('%3.3f',data.k));

data.b=2*data.dampingRatio*data.omega*data.m;
set(handles.bTag,'String',sprintf('%3.3f',data.b));

data.torque = str2num(get(handles.torqueTag,'String'));
data.maxRPM=str2num(get(handles.rpmTag,'String'));
data.maxSimulationTime=str2num(get(handles.simTimeTag,'String'));
data.handles=handles;
data.currentAngle=data.angleInitial;
data.minY=0;
data.maxY=4;


[g_msg,g_status]=nma_lab3_eme_121(data);


% --- Executes on button press in stopTag.
function stopTag_Callback(hObject, eventdata, handles)
% hObject    handle to stopTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

userData = get(handles.figure1, 'UserData');
userData.stop = true;
set(handles.figure1, 'UserData',userData);
%enable_all(handles,'on');



function simTimeTag_Callback(hObject, eventdata, handles)
% hObject    handle to simTimeTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simTimeTag as text
%        str2double(get(hObject,'String')) returns contents of simTimeTag as a double


% --- Executes during object creation, after setting all properties.
function simTimeTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simTimeTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Itag_Callback(hObject, eventdata, handles)
% hObject    handle to Itag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Itag as text
%        str2double(get(hObject,'String')) returns contents of Itag as a double


% --- Executes during object creation, after setting all properties.
function Itag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Itag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pistonMassTag_Callback(hObject, eventdata, handles)
% hObject    handle to pistonMassTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pistonMassTag as text
%        str2double(get(hObject,'String')) returns contents of pistonMassTag as a double


% --- Executes during object creation, after setting all properties.
function pistonMassTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pistonMassTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function pistonFreqTag_Callback(hObject, eventdata, handles)
% hObject    handle to pistonFreqTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pistonFreqTag as text
%        str2double(get(hObject,'String')) returns contents of pistonFreqTag as a double


% --- Executes during object creation, after setting all properties.
function pistonFreqTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pistonFreqTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pistonWnTag_Callback(hObject, eventdata, handles)
% hObject    handle to pistonWnTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pistonWnTag as text
%        str2double(get(hObject,'String')) returns contents of pistonWnTag as a double


% --- Executes during object creation, after setting all properties.
function pistonWnTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pistonWnTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function kTag_Callback(hObject, eventdata, handles)
% hObject    handle to kTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of kTag as text
%        str2double(get(hObject,'String')) returns contents of kTag as a double


% --- Executes during object creation, after setting all properties.
function kTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function bTag_Callback(hObject, eventdata, handles)
% hObject    handle to bTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bTag as text
%        str2double(get(hObject,'String')) returns contents of bTag as a double


% --- Executes during object creation, after setting all properties.
function bTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function [g_msg,g_status]=nma_lab3_eme_121(data)
%
% process function called from GUI driver code to
% plot simulation result
%
% EME 121, spring 2011
%
% by Nasser M. Abbasi, Matlab 2011
% April 2011
%

[xInitial,~]=f(data.L,data.R,data.angleInitial);
data.xInitial=xInitial;
data.xSpeedInitial=0;

data.timeStep=0;
data.solver='ode15s';

g_status = true;
g_msg='';

set(0,'CurrentFigure',data.handles.figure1);
cla(data.handles.xAxesTag,'reset');
cla(data.handles.angleAxesTag,'reset');
cla(data.handles.rpmAxesTag,'reset');
cla(data.handles.mainAxes,'reset');

cla(data.handles.forceAxes,'reset');
cla(data.handles.accAxes,'reset');

IC = [data.angleInitial;data.xInitial;...
```

```
    data.angleSpeedInitial;data.xSpeedInitial];
options = odeset('OutputFcn',@output);

%Check if using auto step size of an explicit step size. see
%GUI for input choice.
if data.timeStep==0
    timeSpan=[0 data.maxSimulationTime];
else
    timeSpan=0:data.timeStep:data.maxSimulationTime;
end

%call  the solver

SCREEN_CAPTURE=false;  %set to zero for animation capture

currentIter=0;

if SCREEN_CAPTURE
    frameNumber = 0;
    actualFrameNumber = 0;
    theFrame=getframe(gcf);
    [im,MAP]=rgb2ind(theFrame.cdata,32,'nodither');
    %MAP=colormap(gray);
end
[t,x] = feval(data.solver,@rhs,timeSpan,IC,options);

if SCREEN_CAPTURE
    imwrite(im,MAP,'demo.gif','DelayTime',0,'LoopCount',inf,...
        'WriteMode','overwrite');
end

%-----------------------------------
% ODE solver function
%-----------------------------------
    function dx=rhs(t,x)
        %the ode45 function

        k=data.k;
        R=data.R;
        L=data.L;
        I=data.I;
        b=data.b;
        m=data.m;
        tao=data.torque;

        [d,ext]=f(L,R,x(1));
        y=sqrt(L^2-R^2*(cos(x(1)))^2);
        if x(3)/(2*pi)*60>data.maxRPM
            data.torque=0;
        end

        %fprintf('angle=%3.3f\t x(2)=%3.3f\t f(theta)=%3.3f\n',mod(x(1)*180/pi,360),x(2),d);

        dx=[x(3);
            x(4);

            tao/I + (k*R/I)*cos(x(1)) * ( 1+R*sin(x(1))/y )* ...
            (-y-R*sin(x(1)) + x(2) ) + ...
            b*R/I * cos(x(1)) * ( 1 + (R*sin(x(1))/y) ) *...
            ( x(4) - R*cos(x(1)) * (1 + (R*sin(x(1))/y) ) * x(3) );
```

```
            k*y/m + k*R*sin(x(1))/m - k*x(2)/m - b*x(4)/m ...
            + b*R*x(3)*cos(x(1))/m + b*x(3)*R^2*cos(x(1))*sin(x(1))/(m*y)
            ];

    end
%---------------------------------
%  ODE output function
%---------------------------------
    function status= output(t,x,~)
        %called by ode45 after each step. Plot the current
        %pendulum position for simulation

        userData = get(data.handles.figure1, 'UserData');
        if userData.stop
            status=true;
            g_status =true;
            return;
        end

        if isempty(t)
            status = true;
        else

            currentIter=currentIter+1;
            if currentIter==1
                status = false;
                return;
            end

            [D,ext]=f(data.L,data.R,x(1,1)) ;

            data.currentAngle=x(1);
            % plot angle vs time

            if currentIter>2
                % position vs. time
                set(0,'CurrentFigure',data.handles.figure1);
                set(data.handles.figure1, 'CurrentAxes',...
                    data.handles.xAxesTag);
                plot([data.lastTime t], [data.lastPosition x(2)],'-');
                xlim([0 data.maxSimulationTime]);
                %                 if abs(data.minY-data.maxY)>10*eps
                %                     ylim([data.minY data.maxY]);
                %                 end
                xlabel('time (sec)');
                title({'Position Vs. time';sprintf('x = %3.7f meter',x(2))});
                hold on;
                set(gca,'FontSize',7);
                drawnow;

                %plot vibration
                set(0,'CurrentFigure',data.handles.figure1);
                set(data.handles.figure1, 'CurrentAxes',...
                    data.handles.angleAxesTag);
                currentSpringExtensionIn_mm = 1000*(x(2)-D);

                plot([data.lastTime t],...
                [data.lastSpringExtension currentSpringExtensionIn_mm],'-');

                %ylabel('mm');
                xlabel('time (sec)');
```

```
title({'virtual spring extension in mm';...
    sprintf('%3.7f',currentSpringExtensionIn_mm)});
hold on;
set(gca,'FontSize',7);
%xlim([0 data.maxSimulationTime]);
drawnow;

% RPMangular speed vs. time plot
set(data.handles.figure1, 'CurrentAxes',...
    data.handles.rpmAxesTag);

plot([data.lastTime t],[data.lastRPM x(3)/(2*pi)*60],'-');
title({'current RPM';sprintf('%3.3f ',x(3)/(2*pi)*60)});
xlim([0 data.maxSimulationTime]);
xlabel('time (sec)');
hold on;
set(gca,'FontSize',7);
drawnow;

%force on piston plot
set(0,'CurrentFigure',data.handles.figure1);
set(data.handles.figure1,'CurrentAxes',data.handles.forceAxes);
currentFd = currentDampingForce(x,data.L,data.R,data.b);
currentFs =  data.k*(x(2)-D);

% if sign(currentFd) ~= sign(currentFs)
%     fprintf('spring force=%3.6f, damping force=%3.6f\n',...
%     currentFs,currentFd);
% end

forceOnPiston = - (currentFs +currentFd);

plot([data.lastTime t], [data.lastTotalForceOnPiston...
    forceOnPiston],'-');

data.lastTotalForceOnPiston=forceOnPiston;

set(gca,'FontSize',7);
title({'force on piston (Newton)';...
    sprintf('%3.7f',forceOnPiston)});
hold on;
xlim([0 data.maxSimulationTime]);
drawnow;

% acc in g
set(data.handles.figure1, 'CurrentAxes',...
    data.handles.accAxes);

currentAcc=(x(4)-data.lastXspeed)/(t-data.lastTime);
currentAcc=currentAcc/9.8;
plot([data.lastTime t],[data.lastAcc currentAcc],'-');
title({'piston acceleration (in g)';...
    sprintf('%3.7f',currentAcc)});

xlim([0 data.maxSimulationTime]);
xlabel('time (sec)');

if currentIter>3
    hold on;
end
set(gca,'FontSize',7);
```

```
    drawnow;

    data.lastTime=t;
    data.lastPosition=x(2);
    data.lastRPM=x(3)/(2*pi)*60;
    data.lastAngle=mod(x(1)*180/pi,360);
    data.lastSringExtension=1000*(x(2)-D);
    data.lastXspeed=x(4);
    data.lastAcc=currentAcc;
else
    data.lastAngle=x(1);
    data.lastPosition=x(2);
    data.lastTime=t;
    data.lastRPM=x(3)/(2*pi)*60;
    data.lastTotalForceOnPiston = - ( data.k*(x(2)-D)+...
        currentDampingForce(x,data.L,data.R,data.b) );
    data.lastSpringExtension=1000*(x(2)-D);
    data.lastXspeed=x(4);
    data.lastAcc=0;
end

%---------------- central simulation now added
set(data.handles.figure1, 'CurrentAxes',data.handles.mainAxes);

%
%             %plot circle
z=-1:.01:1;
plot(data.R*sin(2*pi*z),data.R*cos(2*pi*z),'LineWidth',3);

axis equal;
hold on;
theta = pi/2-x(1);
%
%
%plot R line

plot([0 data.R*cos( theta) D],[0 data.R*sin( theta) 0],'k-');
axis equal;
plot([data.R*cos(theta) D],[data.R*sin(theta) 0],'r-',...
    'LineWidth',2);
plot([0 D],[0 0],'k-');   axis equal;

%plot piston housing
plot([0.8*data.L 1.37*data.L],...
    [data.R/3 data.R/3 ],'-','LineWidth',4,'color','k');

%plot piston housing
plot([1.4*data.L 2*data.L 2*data.L 1.4*data.L],...
    [data.R/3 data.R/3 -data.R/3 -data.R/3],'-',...
    'LineWidth',4,'color','k');

%plot piston housing
plot([0.8*data.L 1.37*data.L],...
    [-data.R/3 -data.R/3 ],'-','LineWidth',4,'color','k');


plot([D x(2,1)],[0 0],'r');   axis equal;
%lengthOfSpring = abs(x(2,1)-D);
%if (x(2,1)-D)<0
%    fprintf('WARNING, solution x is smaller than f(theta)\n');
%end
```

```
plot(data.R*cos( theta),data.R*sin( theta),'o','MarkerSize',6,...
    'MarkerFaceColor','r');   axis equal;
plot(0,0,'o','MarkerSize',4,...
    'MarkerFaceColor','k');   axis equal;

%spring removed for now.
%plot spring
%[xSpring,ySpring]=nma_spring.makeV2(lengthOfSpring,5,data.R/4);
%
%A = [cos(-pi/2) -sin(-pi/2);sin(-pi/2) cos(-pi/2)];
%newSpringCoordinates=arrayfun(@(i) A*[xSpring(i);ySpring(i)],...
%        1:length(xSpring),'UniformOutput',false);
%        newSpringCoordinates= [newSpringCoordinates{:}]';
%
%        xSpring=newSpringCoordinates(:,1);
%        ySpring=newSpringCoordinates(:,2);
%        xSpring=xSpring+(x(2,1)-lengthOfSpring);
%        line(xSpring,ySpring);

%make box
%plot the box
plot(x(2),0,'s','MarkerSize',28,...
    'MarkerFaceColor','m');

set(gca,'XTickLabel',[]);
set(gca,'YTickLabel',[]);
if data.torque>0
    textToShow='torque is on';
else
    textToShow='torque is off';
end

title({'Crank-Piston simulation using Karnopp-Margolis virtual piston mathematical
    'spring/damper attached to piston but not displayed';
    sprintf('Current time = [%4.3f sec], %s',t(1),textToShow)},...
    'FontSize',8);
axis equal;

xlim([-1.3*data.R data.R+2*data.L]);

%draw torque if needed
%plot torque as arc around center.
if abs(data.torque)>10*eps
    n = 100; % The number of points in the arc
    x0=0;y0=0; x1=data.R*.6; y1=0; x2=-data.R*.6; y2=.1;

    P0 = [x0;y0];
    P1 = [x1;y1];
    P2 = [x2;y2];
    v1 = P1-P0;
    v2 = P2-P0;
    c  = det([v1,v2]);
    a  = linspace(0,atan2(abs(c),dot(v1,v2)),n); % Angle range
    v3 = [0,-c;c,0]*v1;
    v  = v1*cos(a)+((norm(v1)/norm(v3))*v3)*sin(a);
    A = [cos(theta) -sin(theta);sin(theta) cos(theta)];
    xy_1=A*[v(1,:)+x0;v(2,:)+y0];
    line(xy_1(1,6:end),xy_1(2,6:end),...
        'LineWidth',1.5,'LineStyle','-',...
        'color','black');
```

```
                line(xy_1(1,1:5),xy_1(2,1:5),'LineWidth',2,...
                    'LineStyle','none',...
                    'Marker','>','MarkerFaceColor','red');
                %text('Interpreter','latex','String','$$\tau$$',...
                %    'Position',[xy_1(1,end),xy_1(2,end)],...
                %    'Fontsize',16);

                hold on;
                axis equal
            end

            axis equal;

            drawnow;
            hold off;
            status = false;
            if SCREEN_CAPTURE
                frameNumber = frameNumber+1;
                if mod(frameNumber,5)==0
                    theFrame = getframe(gcf);
                    actualFrameNumber = actualFrameNumber +1;
                    im(:,:,1,actualFrameNumber ) = ...
                        rgb2ind(theFrame.cdata,MAP,'nodither');
                    %im=rgb2ind(theFrame.cdata,MAP,'nodither');
                    %imwrite(im,MAP,[num2str(actualFrameNumber) '.gif'],'gif');
                end
            end
            %pause(.001);
        end
    end
end
%--------------------------------
% This is the f(theta) function in the textbook
%--------------------------------
function [d,ext]=f(L,R,theta)

ext=R*sin(theta);
d=sqrt(L^2-R^2*cos(theta)^2);
d=d+ext;
end
%------------------
function fd=currentDampingForce(x,L,R,b)
%see equation 6 in report
a=x(1);

fd=b*(x(4)-(R*cos(a)+ R^2*cos(a)*sin(a)/sqrt(L^2-R^2*cos(a)^2))*x(3));

end
classdef nma_spring
    %
    %static class to make spring for plotting animations
    %by Nasser M. Abbasi


    properties
    end

    methods(Static)

        %--------------------------
```

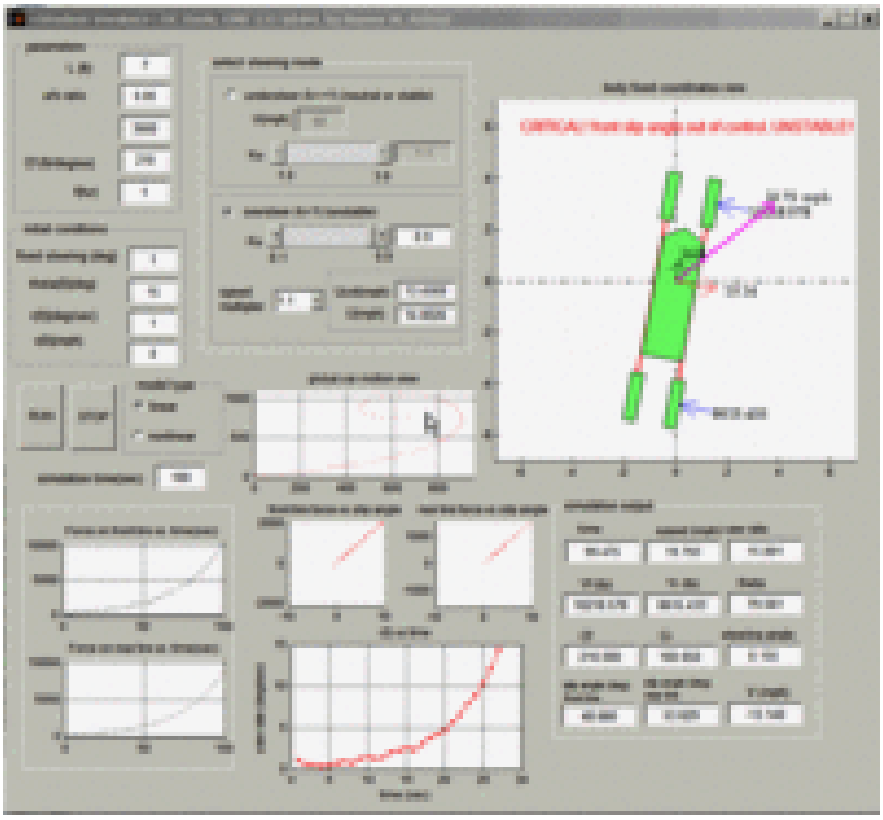
```
%
%---------------------------
function [x,y] = make(r,theta,N)
    %r: total length of spring
    %theta: in radians, anticlock wise is positive,
    %       theta zero is position x-axis
    %N: number of twists in spring
    %
    %OUTPUT:
    % x,y coordinates of line to use to plot spring
    len = (4/6)*r;
    p = zeros(N,2);
    delr = len/N;

    r0     = (1/6)*r;
    p(2,1) = r0;
    p(2,2) = theta;

    for n=3:N-2
        p(n,1)=r0+delr*n;
        z=atan(2*delr/p(n,1));
        p(n,2)=theta+(-1)^n*z;
    end
    p(end-1,1)=(5/6)*r; p(end-1,2)=theta;
    p(end,1)=r; p(end,2)=theta;

    [x,y] = pol2cart(p(:,2),p(:,1));
end

%----------------------------------------
function [x,y] = makeBox(w,h)
    x = zeros(5,1);
    y = zeros(5,1);
    x(1)= w/2; y(1)=0;
    x(2)= w/2; y(2)=h/2;
    x(3)= -w/2; y(3)=h/2;
    x(4)=-w/2;  y(4)=0;
    x(5)=x(1); y(5)=y(1);

end
%---------------------------
%
%---------------------------
function [x,y] = makeV2(len,N,w)
    %len: total length of spring
    %N: number of twists in spring
    %
    %OUTPUT:
    % x,y coordinates of line to use to plot spring

    nCoordinates = 2*N+4;
    p = zeros(nCoordinates,2);
    h = (1/6)*len;
    del=(len-2*h)/(N+1);
    done = false;
    n = 1;
    while not(done)

        if n==1
           p(1,1)=0;
           p(1,2)=0;
```

```
            n = n + 1;
        elseif n==2
            p(2,1)=0;
            p(2,2)=h;
            n = n + 1;
        elseif n<nCoordinates-2
            p(n,1)=w;
            p(n,2)=p(n-1,2)+del;
            n = n  + 1;
            p(n,1)=-w;
            p(n,2)=p(n-1,2);
            n = n + 1;
        elseif n==nCoordinates-1
            p(n,1) = 0;
            p(n,2)=p(n-1,2)+del;
            n = n + 1;
        else
            p(n,1) = 0;
            p(n,2)=p(n-1,2)+h;
            done = true;
        end
    end
    x=p(:,1);
    y=p(:,2);

end



%------------------------
%
%------------------------
function test(theta)

    close all;

    theta = theta*pi/180;
    figure;
    set(gcf,'Units','normalized');
    delr=0.3;
    for r=1:.1:3
        cla;
        [x,y] = nma_spring.make(r,theta,30);
        line(x,y);
        hold on;

        %baseX = [2*delr*cos((pi/2)-theta) -2*delr*sin(theta)];
        %baseY = [-2*delr*sin((pi/2)-theta) 2*delr*cos(theta)];

        baseX = [-2*delr*sin(abs(theta)) 2*delr*sin(abs(theta))];
        baseY = [-2*delr*cos(theta)      2*delr*cos(theta)];
        line(baseX,baseY);

        triX=[baseX(1) ...
            baseX(1) ...
            30*delr*cos(theta)-abs(baseX(1))...
            baseX(1)];

        triY=[baseY(1) ...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
            -(30*delr*sin(abs(theta))+abs(baseY(1)))...
```

```
            baseY(1)];

        line(triX,triY);

        L=sqrt(x(end)^2+y(end)^2);
        xx0=L*cos(theta);
        yy0=-L*sin(abs(theta));
        massX=[xx0 ...
            xx0+2*delr*sin(abs(theta)) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta) ...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-4*delr*sin(abs(theta))...
            xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-4*delr*sin(abs(theta))-4*delr
            xx0];

        massY=[ yy0 ...
            yy0+2*delr*cos(theta) ...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta)) ...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-4*delr*cos(theta)...
            yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-4*delr*cos(theta)+4*delr*sin(
            yy0];

        line(massX,massY);

        xlim([-3,10]); ylim([-10,3]);

        set(gca,'DataAspectRatioMode','manual','DataAspectRatio',[1 1 1]);
        drawnow;
        pause(.1);
    end

end

%-------------------------------
%
%-------------------------------
function testV2(len)

    close all;

    theta = 0;
    figure;
    set(gcf,'Units','normalized');
    delr=0.3;
    for r=1:len/3:len
        cla;
        [x,y] = nma_spring.makeV2(r,10,1);
        line(x,y);
        hold on;

        %baseX = [2*delr*cos((pi/2)-theta) -2*delr*sin(theta)];
        %baseY = [-2*delr*sin((pi/2)-theta) 2*delr*cos(theta)];

        baseX = [-2*delr*sin(abs(theta)) 2*delr*sin(abs(theta))];
        baseY = [-2*delr*cos(theta)      2*delr*cos(theta)];
        line(baseX,baseY);

        triX=[baseX(1) ...
            baseX(1) ...
            30*delr*cos(theta)-abs(baseX(1))...
            baseX(1)];
```

```
                triY=[baseY(1) ...
                    -(30*delr*sin(abs(theta))+abs(baseY(1)))...
                    -(30*delr*sin(abs(theta))+abs(baseY(1)))...
                    baseY(1)];

                line(triX,triY);

                L=sqrt(x(end)^2+y(end)^2);
                xx0=L*cos(theta);
                yy0=-L*sin(abs(theta));
                massX=[xx0 ...
                    xx0+2*delr*sin(abs(theta)) ...
                    xx0+2*delr*sin(abs(theta))+4*delr*cos(theta) ...
                    xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-4*delr*sin(abs(theta))...
                    xx0+2*delr*sin(abs(theta))+4*delr*cos(theta)-4*delr*sin(abs(theta))-4*de
                    xx0];

                massY=[ yy0 ...
                    yy0+2*delr*cos(theta) ...
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta)) ...
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-4*delr*cos(theta)...
                    yy0+2*delr*cos(theta)-4*delr*sin(abs(theta))-4*delr*cos(theta)+4*delr*si
                    yy0];

                line(massX,massY);

                xlim([-3,10]); ylim([-10,3]);

                set(gca,'DataAspectRatioMode','manual','DataAspectRatio',[1 1 1]);
                drawnow;
                pause(.3);
            end

        end

    end
end
```

4.4 Lab 4, Simulation of half car, stability



4.4.1 Animation

The GUI based simulation was written in Matlab 2011a. A number of animated GIF files were made. In the table below, clicking on any image will start a gif animation in your browser. The animation will only run once. To run it again, please reload the web page using the reload button on the browser.

These are relatively large animation GIF files and might take 1-2 seconds to load depending on network bandwidth. To run them locally, they can be downloaded as well.

4.4.2 Derivation of equations of motion

The following is the overall diagram of the half car model showing the body fixed speeds and coordinates in relation to the inertial frame of reference X, Y.

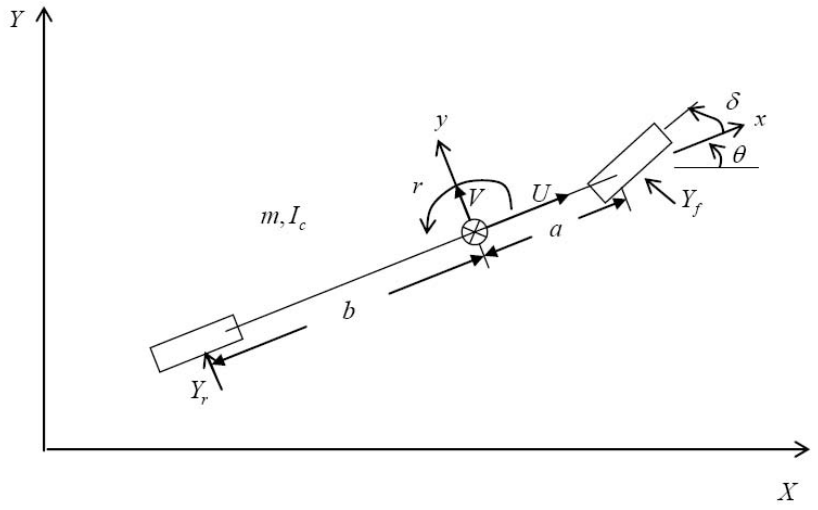


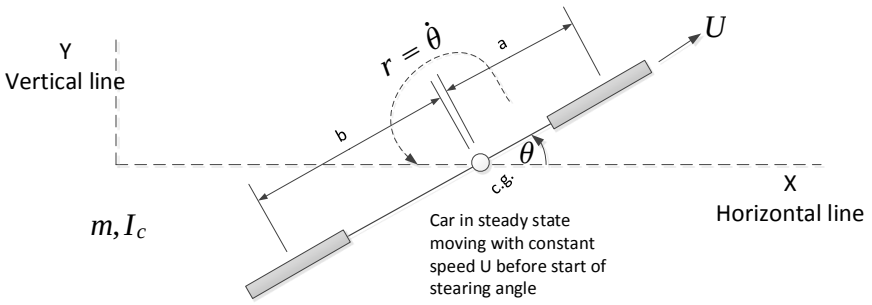
Figure 1 Schematic of the vehicle system

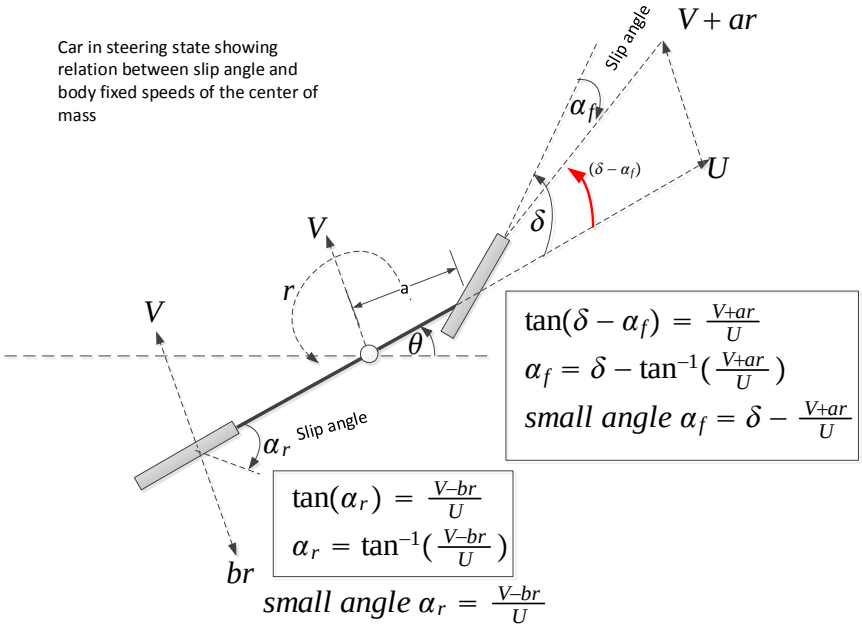
The half car simulation mathematical model equations of motion will now be derived from first principles. The first step is to obtain an expression for the slip angle in terms of the speed of the center of mass of the car in body fixed coordinates. Once the slip angles expression are obtain, then forces are expressed in terms of these angles using the cornering coefficient.

Once the force on each tire is found, then $F = ma$ is applied and hence the equation of motion of the center of mass of the car is obtained.

The slip angle is defined such that it opposes the force from the ground on the tire. Here we have the forces as positive in the vertical direction. Hence the slip angles will grow in the clockwise direction as shown below.

The following diagram shows the derivation of the expressions for slip angles α_f and α_r for the front and rear tires





Hence from the above diagram we have the following 2 equations

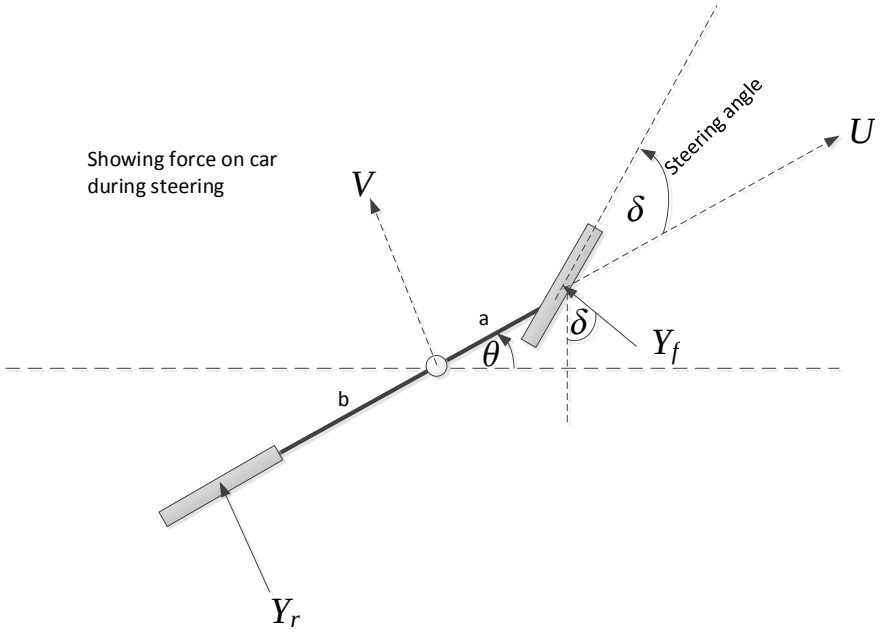
$$\alpha_f(t) = \delta(t) - \tan^{-1}\left(\frac{V(t) + ar(t)}{U}\right)$$
$$\alpha_r(t) = \tan^{-1}\left(\frac{br(t) - V(t)}{U}\right)$$

Now, we apply the cornering formula. This formula relates the force on the tire to the slip angles. Hence

$$Y_f(t) = C_f\alpha_f(t)$$
$$Y_r(t) = C_r\alpha_r(t)$$

The formula $Y = C\alpha$ is similar to the relation between the force and displacement in the spring $F = k\Delta$. Hence C_r and C_f are like stiffness parameters for the tire. The larger these parameters are, the larger the force on the tire will have to be to generate the same slip angle. The units of C_r and C_f are N/rad.

Now we apply $F = ma$ to the lateral direction only, since we assume speed U is constant and hence no force in the longitudinal direction. We also apply moment equation around the c.g. of the car.



This results in the following 2 equations

$$m\left(\dot{V} + rU\right) = Y_r + Y_f \cos\delta(t)$$

And the moment equation gives

$$I_z\dot{r} = Y_f \cos\delta - bY_r$$

By solving for $\dot{V}$ and $\dot{r}$ from the above we obtain

$$\dot{V} = \frac{Y_f \cos\delta(t)}{m} + \frac{Y_r}{m} - rU$$
$$\dot{r} = \frac{Y_f \cos\delta(t)}{I_z} - \frac{bY_r}{I_z}$$

Finally, to obtain the trajectory of the car, we transform back to inertial coordinates using the following 2 equations

$$\dot{\theta}(t) = r(t)$$
$$\dot{X}(t) = U\cos\theta(t) - V(t)\sin\theta(t)$$
$$\dot{Y}(t) = U\sin\theta(t) + V(t)\cos\theta(t)$$

The above equations are solved and the result displayed during simulation as described below

Description of parameters used in the equations of motion

This table below is a description of each term in the above equations and all other parameters and variables used in the simulation

term	meaning
$\alpha_f(t)$	slip angle at the front tire
$\alpha_r(t)$	slip angle at the rear tire
$Y_f(t)$	Lateral force at the front tire
$Y_r(t)$	Lateral force at the front tire
$\delta(t)$	steering angle $\delta(t) = \delta_0 \sin(2\pi f t)$
f	steering angle frequency used in finding $\delta(t)$ in Hz (user input)
δ_0	amplitude of $\delta(t)$ in radians (user input)
U	forward speed on center of car. Constant. (user input when understeer else $U = \lambda U_{crit}$)
λ	speed multiplier to multiply $U_{cirtial}$ with for the case when oversteer mode. (user input 0.9, 1, 1.1)
$r(t)$	yaw rate
$\theta(t)$	angle car makes with inertial X axis. $r = \frac{d\theta}{dt}$
$X(t)$	global X coordinate of center of car
$Y(t)$	global Y coordinate of center of car
C_f	Corner coefficient of front tire (user input)
C_r	Corner coefficient of rear tire $C_r = k_u \frac{a}{b} C_f$
m	mass of car (user input)
a	distance from c.g. of car to front tire (user input)
b	distance from c.g. of car to the rear tire. (user input)
k_u	steering mode parameter. $k_u > 1$ means understeer, else oversteer (user input 1.1, 1, 0.9)
$U_{critical}$	Critical speed over which car becomes unstable in oversteer mode. $U_{crit}^2 = \frac{(a+b)^2 C_f C_r}{m(aC_f - bC_r)}$
I_z	moment of inertia of car around axis through its c.g. $I_z = \frac{mab}{2}$

Linear equations of motion

The following is the list of the linear equations that represent the dynamics of the car

$$
\begin{aligned}
\delta(t) &= \delta_0 \sin(2\pi f t) \\
\alpha_f(t) &= \delta(t) - \frac{V(t) + a\, r(t)}{U} \\
\alpha_r(t) &= \frac{b\, r(t) - V(t)}{U} \\
Y_f(t) &= C_f \alpha_f(t) \\
Y_r(t) &= C_r \alpha_r(t) \\
\dot{V}(t) &= \frac{Y_f(t)}{m} + \frac{Y_r(t)}{m} - U r(t) \\
\dot{r}(t) &= \frac{a Y_f(t)}{I_z} - \frac{Y_r(t) b}{I_z} \\
\dot{\theta}(t) &= r(t) \\
\dot{X}(t) &= U \cos\theta(t) - V(t) \sin\theta(t) \\
\dot{Y}(t) &= U \sin\theta(t) + V(t) \cos\theta(t)
\end{aligned}
$$

Non-linear equations of motion

The following is the list of nonlinear equations that represent the dynamics of the car. Both the linear and non-linear equations are valid for small angles, but the equations below produces a more accurate result for small angles.

$$
\begin{aligned}
\delta(t) &= \delta_0 \sin(2\pi f t) \\
\alpha_f(t) &= \delta(t) - \tan^{-1}\left(\frac{V(t) + a\, r(t)}{U}\right) \\
\alpha_r(t) &= \tan^{-1}\left(\frac{b\, r(t) - V(t)}{U}\right) \\
Y_f(t) &= C_f \alpha_f(t) \\
Y_r(t) &= C_r \alpha_r(t) \\
\dot{V}(t) &= \frac{Y_f(t)}{m} \cos\delta(t) + \frac{Y_r(t)}{m} - U r(t) \\
\dot{r}(t) &= \frac{a Y_f(t) \cos\delta(t)}{I_z} - \frac{Y_r(t) b}{I_z} \\
\dot{\theta}(t) &= r(t) \\
\dot{X}(t) &= U \cos\theta(t) - V(t) \sin\theta(t) \\
\dot{Y}(t) &= U \sin\theta(t) + V(t) \cos\theta(t)
\end{aligned}
$$

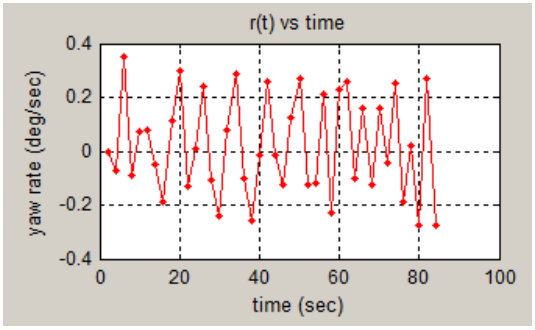
4.4.3 Results of simulation

Unstable $ku = 0.9$ results

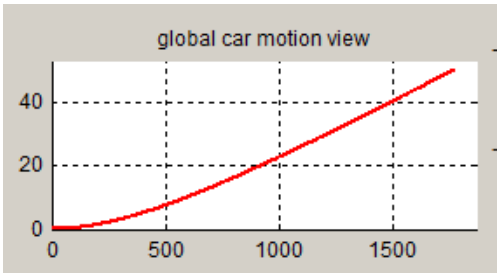
Critical speed below car speed (multiplier = 0.9)

Ku UNSTABLE (0.9)
Speed multiplier = 0.9
Simulation time = 100 seconds

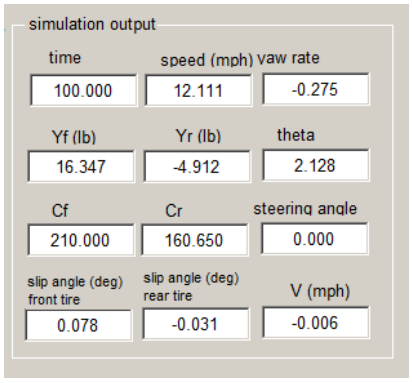
U=12.111 MPh
U Critical = 12.46 Mph



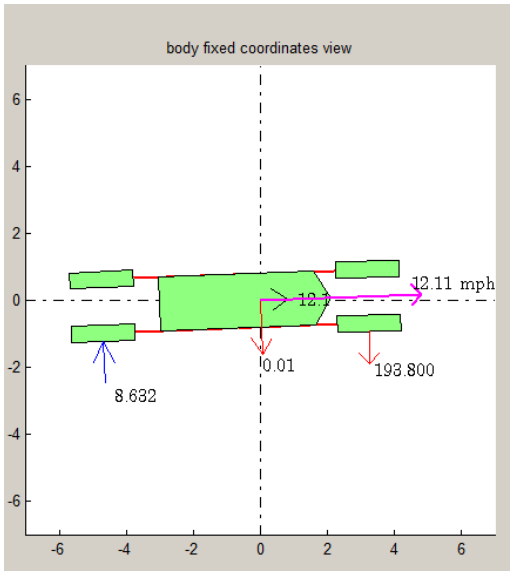
Yaw rate plot



Trajectory plot



Final simulation state

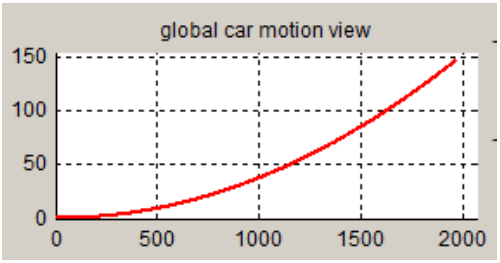


Final fixed-body coordinates state

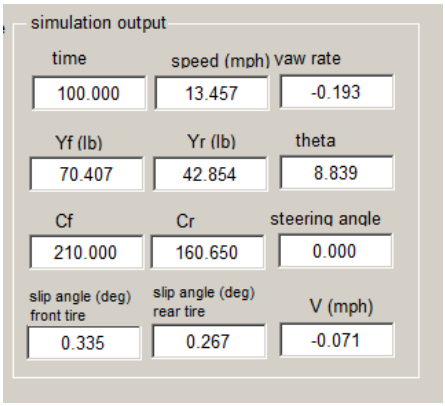
Critical speed the same as car speed (multiplier = 1)

Ku UNSTABLE (0.9)
Speed multiplier = 1.0
Simulation time = 100 seconds

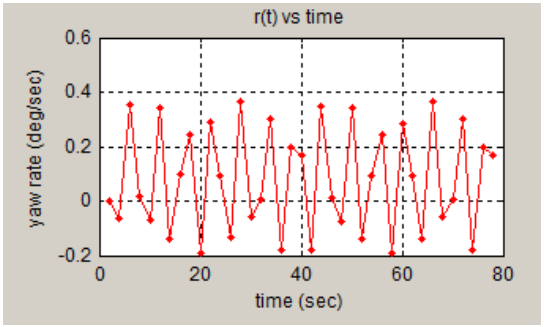
U=13.46 mph
U Critical =13.46 mph



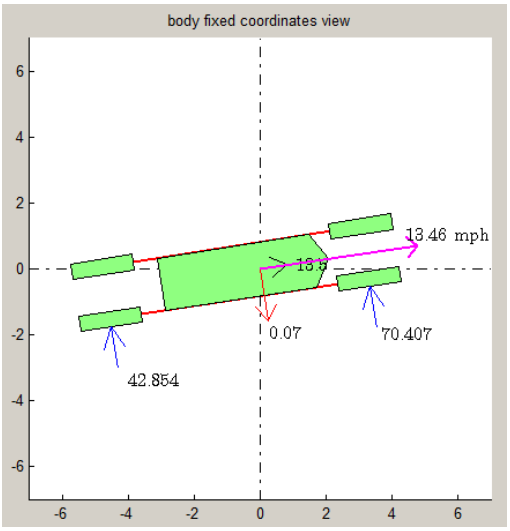
Trajectory plot



Final simulation state



Yaw rate plot

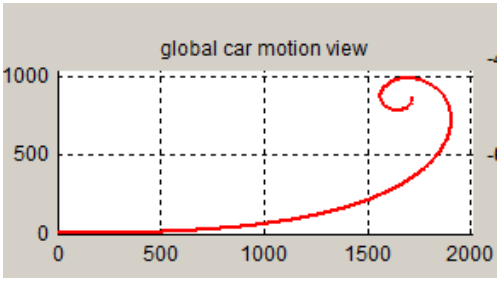


Final fixed-body coordinates state

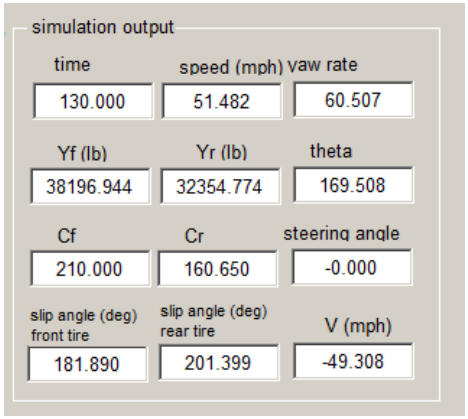
Critical speed larger than speed (multiplier = 1.1)

Ku UNSTABLE (0.9)
Speed multiplier = 1.1
Simulation time = 100 seconds

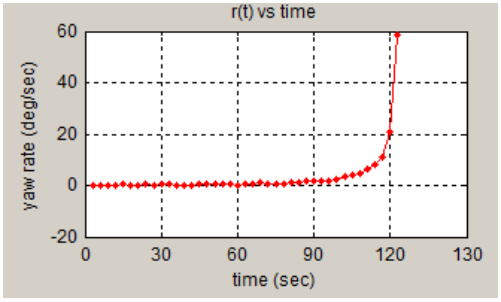
U=13.46 mph
U Critical =14.80 mph



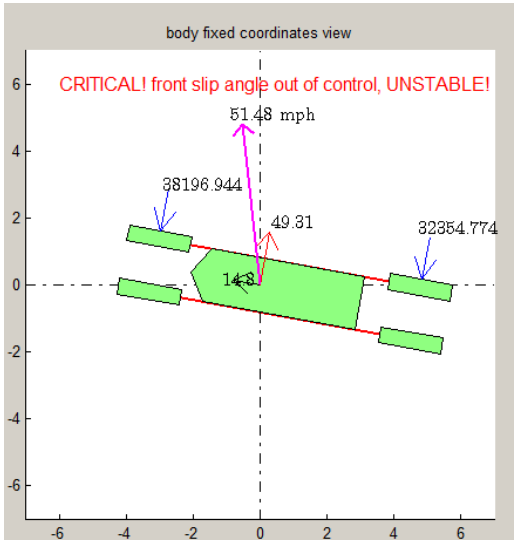
Trajectory plot



Final simulation state



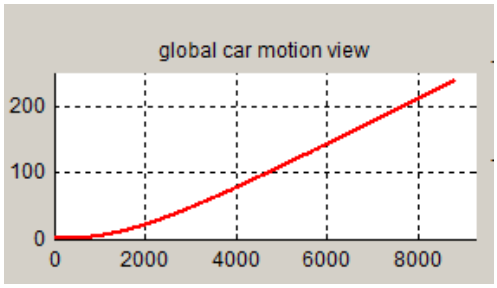
Yaw rate plot



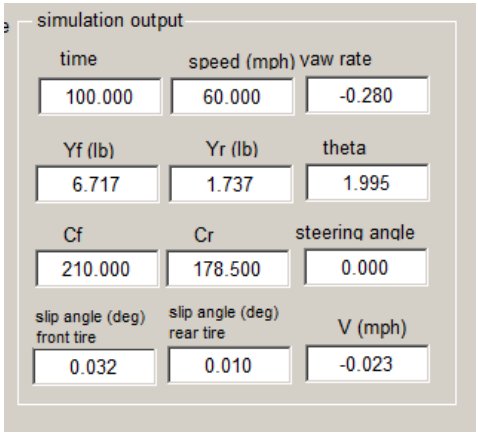
Final fixed-body coordinates state

Neutral $ku = 1$ results (60 Mph)

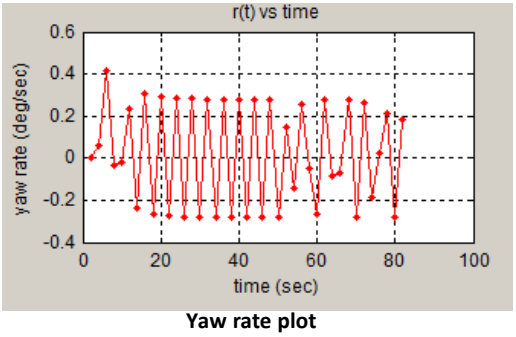
Ku neutral (1.0)
Speed = 60 Mph
Simulation time = 100 seconds



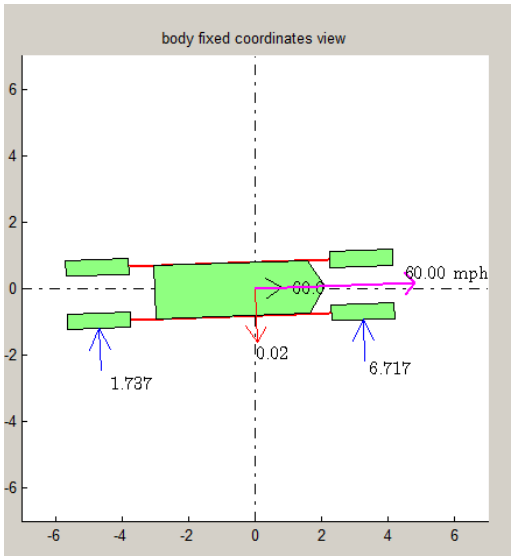
Trajectory plot



Final simulation state



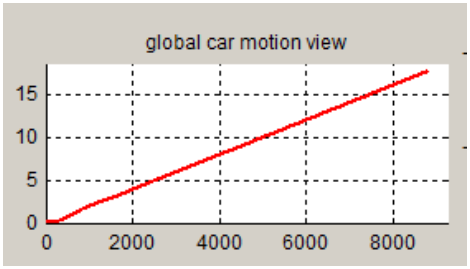
Yaw rate plot



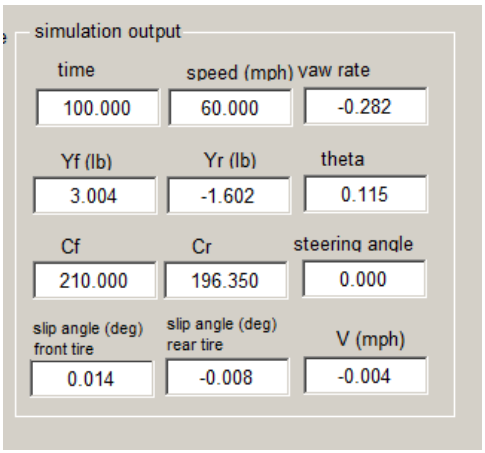
Final fixed-body coordinates state

Stable $K_u = 1.1$ results

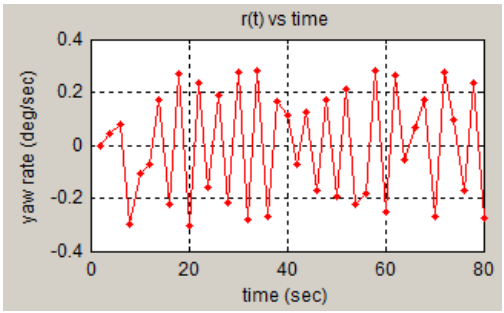
Ku stable (1.1)
Speed = 60 Mph
Simulation time = 100 seconds



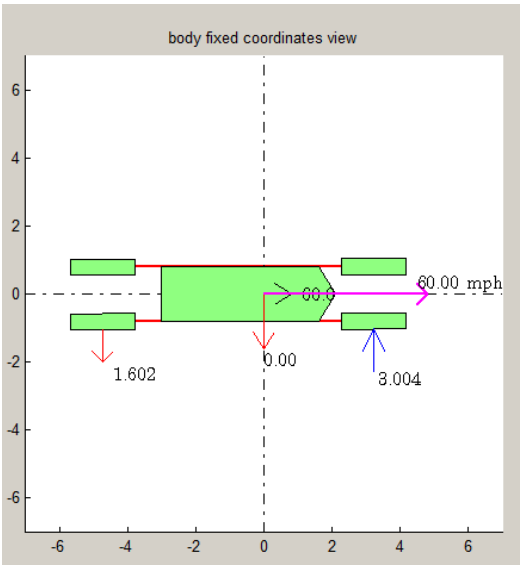
Trajectory plot



Final simulation state



Yaw rate plot



Final fixed-body coordinates state

4.4.4 Discussion of simulation results

The following summarizes the results observed from the simulation.

$ku = 0.9$ Unstable (oversteer)

1. critical speed below U (0.9): Even though ku used was unstable, the car did not become unstable since the speed remained below critical speed. The car remained in an almost straight line, and the yaw rate r did not grow.

2. critical speed the same as U (1.0): The car also remained stable, the yaw rate did not grow, the trajectory has a little more curvature than the first case but pretty much remained almost straight.

3. critical speed larger than speed U (1.1): The car became unstable. As now an eigenvalue is on the right side of the complex plane. The yaw rate was seen to grow with time, and the car trajectory

shows that the car went into cycles with increasing speed.

$Ku = 1.0$ (Neutral) understeer

The car speed used was 60m.p.h. The car remained stable and the yaw rate was sinusoidal but did not grow up in magnitude. The trajectory shows the car remained in an almost straight line.

$Ku = 1.1$ understeer

The car speed used was 60m.p.h. Also in this case the car remained stable and the yaw rate did not grow up in magnitude. The trajectory shows the car remained in exactly straight line during all the simulation time.

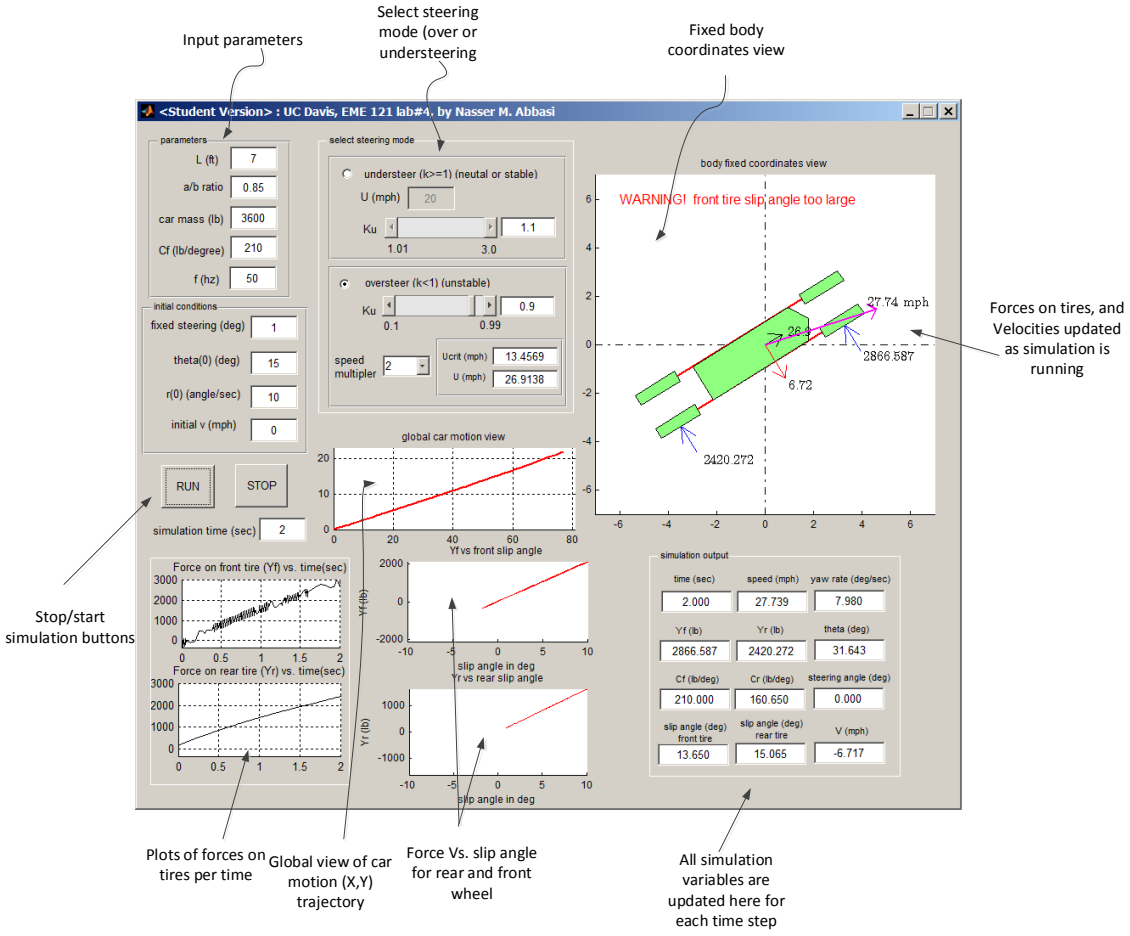
Conclusion

The mathematical model used predicted that when the car speed used is larger than a critical speed value, an eigenvalue will become unstable when $k_u < 1$. Hence the car would become unstable. The above result confirmed.

The model also predicted that when $k_u \geq 1$, then the car will remain stable for any U. And this was also confirmed by showing that the car was stable for $k_u = 1$ and $k_u = 1.1$ for different speed. This report shows the case for $U = 60mph$, but all other speeds tried generated similar results and the car remained stable as long as $k_u \geq 1$.

4.4.5 Appendix

Description of user interface used for simulation



Source code

Listing 4.1: lab4.m

```
function varargout = lab4(varargin)
% LAB4 MATLAB code for lab4.fig
%      LAB4, by itself, creates a new LAB4 or raises the existing
%      singleton*.
%
%      H = LAB4 returns the handle to a new LAB4 or the handle to
%      the existing singleton*.
%
%      LAB4('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LAB4.M with the given input arguments.
%
%      LAB4('Property','Value',...) creates a new LAB4 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before lab4_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to lab4_OpeningFcn via varargin.
%
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% GUI file for my lab4 assignment, UC davis, spring 2011 EME121
% by Nasser M. Abbasi
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help lab4

% Last Modified by GUIDE v2.5 17-May-2011 02:51:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @lab4_OpeningFcn, ...
                   'gui_OutputFcn',  @lab4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before lab4 is made visible.
function lab4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to lab4 (see VARARGIN)

% Choose default command line output for lab4
handles.output = hObject;

%nma_set_figure_position(handles.figure1,0.1,0.1,.75,.79);

set(handles.figure1, 'UserData',[]);
set(handles.figure1,'Name','UC Davis, EME 121 lab#4, by Nasser M. Abbasi');

userData.stop = false;
set(handles.figure1, 'UserData',userData);

contents = cellstr(get(handles.criticalMultiplierTag,'String'));
criticalMultipler=str2num(contents{get(handles.criticalMultiplierTag,...
    'Value')});
updateUnstable(criticalMultipler,handles);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes lab4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = lab4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function lenTag_Callback(hObject, eventdata, handles)
% hObject    handle to lenTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of lenTag as text
%        str2double(get(hObject,'String')) returns contents of lenTag as a double


% --- Executes during object creation, after setting all properties.
function lenTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lenTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function abRatioTag_Callback(hObject, eventdata, handles)
% hObject    handle to abRatioTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of abRatioTag as text
%        str2double(get(hObject,'String')) returns contents of abRatioTag as a double


% --- Executes during object creation, after setting all properties.
function abRatioTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to abRatioTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function mTag_Callback(hObject, eventdata, handles)
% hObject    handle to mTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mTag as text
%        str2double(get(hObject,'String')) returns contents of mTag as a double


% --- Executes during object creation, after setting all properties.
function mTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CFTag_Callback(hObject, eventdata, handles)
% hObject    handle to CFTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CFTag as text
%        str2double(get(hObject,'String')) returns contents of CFTag as a double


% --- Executes during object creation, after setting all properties.
function CFTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CFTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function kTag_Callback(hObject, eventdata, handles)
% hObject    handle to kTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of kTag as text
%        str2double(get(hObject,'String')) returns contents of kTag as a double


% --- Executes during object creation, after setting all properties.
function kTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function fTag_Callback(hObject, eventdata, handles)
% hObject    handle to fTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fTag as text
%        str2double(get(hObject,'String')) returns contents of fTag as a double


% --- Executes during object creation, after setting all properties.
function fTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function UTag_Callback(hObject, eventdata, handles)
% hObject    handle to UTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of UTag as text
%        str2double(get(hObject,'String')) returns contents of UTag as a double


% --- Executes during object creation, after setting all properties.
function UTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to UTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function deltaZeroTag_Callback(hObject, eventdata, handles)
% hObject    handle to deltaZeroTag (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of deltaZeroTag as text
%        str2double(get(hObject,'String')) returns contents of deltaZeroTag as a double


% --- Executes during object creation, after setting all properties.
function deltaZeroTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to deltaZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function thetaZeroTag_Callback(hObject, eventdata, handles)
% hObject    handle to thetaZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of thetaZeroTag as text
%        str2double(get(hObject,'String')) returns contents of thetaZeroTag as a double


% --- Executes during object creation, after setting all properties.
function thetaZeroTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to thetaZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function rZeroTag_Callback(hObject, eventdata, handles)
% hObject    handle to rZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rZeroTag as text
%        str2double(get(hObject,'String')) returns contents of rZeroTag as a double


% --- Executes during object creation, after setting all properties.
function rZeroTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function vZeroTag_Callback(hObject, eventdata, handles)
% hObject    handle to vZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vZeroTag as text
%        str2double(get(hObject,'String')) returns contents of vZeroTag as a double


% --- Executes during object creation, after setting all properties.
function vZeroTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vZeroTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function maxtTag_Callback(hObject, eventdata, handles)
% hObject    handle to maxtTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of maxtTag as text
%        str2double(get(hObject,'String')) returns contents of maxtTag as a double


% --- Executes during object creation, after setting all properties.
function maxtTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxtTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in runTag.
function runTag_Callback(hObject, eventdata, handles)
% hObject    handle to runTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
userData = get(handles.figure1, 'UserData');
userData.stop = false;
set(handles.figure1, 'UserData',userData);

data.L=str2num(get(handles.lenTag,'String'));
data.abRatio=str2num(get(handles.abRatioTag,'String'));
data.m=str2num(get(handles.mTag,'String'));
data.Cf=str2num(get(handles.CFTag,'String'));
data.Cf=data.Cf*180/pi; %read as lb/degree, convert to lb/radian

data.f=str2num(get(handles.fTag,'String'));

data.b= data.L/(1+data.abRatio);
data.a= data.L-data.b;
data.Ic = data.m*data.a*data.b/2;

contents = cellstr(get(handles.criticalMultiplierTag,'String'));
data.criticalMultipler=str2num(contents{get(...
    handles.criticalMultiplierTag,'Value')});

MPH=(60*60)/5280;
FTS=1/MPH;
if get(handles.underSteerBtn,'Value')==1
   data.ku=get(handles.kuStableTag,'Value');
   data.U=str2num(get(handles.UTag,'String'));
   data.U=data.U*FTS;  %ft/sec
   data.Cr = data.ku*(data.a/data.b)*data.Cf;

else
   data.ku=get(handles.unstableKuTag,'Value');
   data.Cr = data.ku*(data.a/data.b)*data.Cf;
   data.Ucrit = sqrt( (data.a+data.b)^2*...
       (data.Cf*data.Cr)/(data.m*(data.a*data.Cf-data.b*data.Cr)));
   data.U=data.criticalMultipler*data.Ucrit ;
   set(handles.calculatedUcritTag,'String', data.Ucrit*MPH);
   set(handles.calcuatedUTag,'String', data.U*MPH);
end

data.deltaZero=str2num(get(handles.deltaZeroTag,'String'));
data.deltaZero=data.deltaZero*pi/180;

data.thetaZero=str2num(get(handles.thetaZeroTag,'String'));
data.thetaZero=data.thetaZero*pi/180;

data.rZero=str2num(get(handles.rZeroTag,'String'));
data.rZero=data.rZero*pi/180;

data.vZero=str2num(get(handles.vZeroTag,'String'));
data.vZero=data.vZero*FTS;

data.maxt=str2num(get(handles.maxtTag,'String'));

data.handles=handles;

enableAll(handles,'off')
[g_msg,g_status]=lab4Main(data);
enableAll(handles,'on');

% --- Executes on button press in stopTag.
function stopTag_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to stopTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

userData = get(handles.figure1, 'UserData');
userData.stop = true;
set(handles.figure1, 'UserData',userData);



% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over underSteerBtn.
function underSteerBtn_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to underSteerBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
i=1;

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over overSteerBtn.
function overSteerBtn_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to overSteerBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
i=1;


% --- Executes when selected object is changed in steeringModePanel.
function steeringModePanel_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in steeringModePanel
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%       EventName: string 'SelectionChanged' (read only)
%       OldValue: handle of the previously selected object or empty if none was selected
%       NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(hObject,'Tag')
    case 'underSteerBtn'
          set(handles.unstableKuTag,'Enable','off');
          set(handles.criticalMultiplierTag,'Enable','off');
          set(handles.unstableKuValueTag,'Enable','off');
          set(handles.calculatedUcritTag,'Enable','off');
          set(handles.calcuatedUTag,'Enable','off');

          set(handles.kuStableTag,'Enable','on');
          set(handles.UTag,'Enable','on');
          set(handles.stableKuValueTag,'Enable','inactive');

    case 'overSteerBtn'
 set(handles.unstableKuTag,'Enable','inactive');

          set(handles.unstableKuValueTag,'Enable','inactive');
          set(handles.calculatedUcritTag,'Enable','inactive');
          set(handles.calcuatedUTag,'Enable','inactive');

          set(handles.unstableKuTag,'Enable','on');
          set(handles.criticalMultiplierTag,'Enable','on');

          set(handles.kuStableTag,'Enable','off');
          set(handles.UTag,'Enable','off');
          set(handles.stableKuValueTag,'Enable','off');
end
```

```
% --------------------------------------------------------------------
function steeringModePanel_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to steeringModePanel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on selection change in criticalMultiplierTag.
function criticalMultiplierTag_Callback(hObject, eventdata, handles)
% hObject    handle to criticalMultiplierTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns criticalMultiplierTag contents as
%     contents{get(hObject,'Value')} returns selected item from criticalMultiplierTag

contents = cellstr(get(hObject,'String'));
criticalMultipler=str2num(contents{get(hObject,'Value')});

updateUnstable(criticalMultipler,handles);




% --- Executes during object creation, after setting all properties.
function criticalMultiplierTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to criticalMultiplierTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function stableKuSliderTag_Callback(hObject, eventdata, handles)
% hObject    handle to stableKuSliderTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function stableKuSliderTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to stableKuSliderTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Executes on slider movement.
function kuStableTag_Callback(hObject, eventdata, handles)
% hObject    handle to kuStableTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

v=get(hObject,'Value');
set(handles.stableKuValueTag,'String',v);



% --- Executes during object creation, after setting all properties.
function kuStableTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kuStableTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function unstableKuTag_Callback(hObject, eventdata, handles)
% hObject    handle to unstableKuTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

v=get(hObject,'Value');
set(handles.unstableKuValueTag,'String',v);

contents = cellstr(get(handles.criticalMultiplierTag,'String'));
criticalMultipler=...
    str2num(contents{get(handles.criticalMultiplierTag,'Value')});
updateUnstable(criticalMultipler,handles);

% --- Executes during object creation, after setting all properties.
function unstableKuTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to unstableKuTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function stableKuValueTag_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to stableKuValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of stableKuValueTag as text
%        str2double(get(hObject,'String')) returns contents of stableKuValueTag as a double


% --- Executes during object creation, after setting all properties.
function stableKuValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to stableKuValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function unstableKuValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to unstableKuValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of unstableKuValueTag as text
%        str2double(get(hObject,'String')) returns contents of unstableKuValueTag as a doubl


% --- Executes during object creation, after setting all properties.
function unstableKuValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to unstableKuValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function calculatedUcritTag_Callback(hObject, eventdata, handles)
% hObject    handle to calculatedUcritTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of calculatedUcritTag as text
%        str2double(get(hObject,'String')) returns contents of calculatedUcritTag as a doubl


% --- Executes during object creation, after setting all properties.
function calculatedUcritTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to calculatedUcritTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function calcuatedUTag_Callback(hObject, eventdata, handles)
% hObject    handle to calcuatedUTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of calcuatedUTag as text
%        str2double(get(hObject,'String')) returns contents of calcuatedUTag as a double


% --- Executes during object creation, after setting all properties.
function calcuatedUTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to calcuatedUTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function updateUnstable(criticalMultipler,handles)

MPH=(60*60)/5280;
FTS=1/MPH;

L       = str2num(get(handles.lenTag,'String'));
abRatio = str2num(get(handles.abRatioTag,'String'));
m       = str2num(get(handles.mTag,'String'));
Cf      = str2num(get(handles.CFTag,'String'));
Cf      = Cf*180/pi;

b       = L/(1+abRatio);
a       = L-b;

ku      = get(handles.unstableKuTag,'Value');

Cr      = ku*(a/b)*Cf;

Ucrit   = sqrt( (a+b)^2*(Cf*Cr)/(m*(a*Cf-b*Cr)));
U       = criticalMultipler*Ucrit ;

set(handles.calculatedUcritTag,'String',Ucrit*MPH);
set(handles.calcuatedUTag,'String', U*MPH);

function timeValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to timeValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of timeValueTag as text
%        str2double(get(hObject,'String')) returns contents of timeValueTag as a double


% --- Executes during object creation, after setting all properties.
function timeValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to timeValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function speedValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to speedValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of speedValueTag as text
%        str2double(get(hObject,'String')) returns contents of speedValueTag as a double


% --- Executes during object creation, after setting all properties.
function speedValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to speedValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function yawRateValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to yawRateValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of yawRateValueTag as text
%        str2double(get(hObject,'String')) returns contents of yawRateValueTag as a double


% --- Executes during object creation, after setting all properties.
function yawRateValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yawRateValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function YfValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to YfValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of YfValueTag as text
%        str2double(get(hObject,'String')) returns contents of YfValueTag as a double


% --- Executes during object creation, after setting all properties.
function YfValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to YfValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function YrValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to YrValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of YrValueTag as text
%        str2double(get(hObject,'String')) returns contents of YrValueTag as a double


% --- Executes during object creation, after setting all properties.
function YrValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to YrValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function thetaValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to thetaValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of thetaValueTag as text
%        str2double(get(hObject,'String')) returns contents of thetaValueTag as a double
```

```
% --- Executes during object creation, after setting all properties.
function thetaValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to thetaValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CfValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to CfValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CfValueTag as text
%        str2double(get(hObject,'String')) returns contents of CfValueTag as a double


% --- Executes during object creation, after setting all properties.
function CfValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CfValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CrValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to CrValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CrValueTag as text
%        str2double(get(hObject,'String')) returns contents of CrValueTag as a double


% --- Executes during object creation, after setting all properties.
function CrValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CrValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function steerAngleValueTag_Callback(hObject, eventdata, handles)
% hObject    handle to steerAngleValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of steerAngleValueTag as text
%        str2double(get(hObject,'String')) returns contents of steerAngleValueTag as a double


% --- Executes during object creation, after setting all properties.
function steerAngleValueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to steerAngleValueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function alpha_r_valueTag_Callback(hObject, eventdata, handles)
% hObject    handle to alpha_r_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha_r_valueTag as text
%        str2double(get(hObject,'String')) returns contents of alpha_r_valueTag as a double


% --- Executes during object creation, after setting all properties.
function alpha_r_valueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha_r_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function alpha_f_valueTag_Callback(hObject, eventdata, handles)
% hObject    handle to alpha_f_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha_f_valueTag as text
%        str2double(get(hObject,'String')) returns contents of alpha_f_valueTag as a double


% --- Executes during object creation, after setting all properties.
function alpha_f_valueTag_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to alpha_f_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_valueTag_Callback(hObject, eventdata, handles)
% hObject    handle to V_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_valueTag as text
%        str2double(get(hObject,'String')) returns contents of V_valueTag as a double


% --- Executes during object creation, after setting all properties.
function V_valueTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_valueTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function enableAll(handles,to)

set(handles.lenTag,'Enable',to);
set(handles.abRatioTag,'Enable',to);
set(handles.mTag,'Enable',to);
set(handles.CFTag,'Enable',to);
set(handles.fTag,'Enable',to);
set(handles.deltaZeroTag,'Enable',to);
set(handles.thetaZeroTag,'Enable',to);
set(handles.rZeroTag,'Enable',to);
set(handles.vZeroTag,'Enable',to);
set(handles.maxtTag,'Enable',to);
set(handles.underSteerBtn,'Enable',to);

if strcmp(to,'on')
    if get(handles.underSteerBtn,'Value')==1
        set(handles.kuStableTag,'Enable',to);
        set(handles.UTag,'Enable',to);
        set(handles.stableKuValueTag,'Enable','inactive');
    end
else
   set(handles.kuStableTag,'Enable',to);
   set(handles.UTag,'Enable',to);
   set(handles.stableKuValueTag,'Enable','off');
end
```

```
set(handles.overSteerBtn,'Enable',to);
if strcmp(to,'on')
    if get(handles.overSteerBtn,'Value')==1
       set(handles.unstableKuTag,'Enable',to);
       set(handles.criticalMultiplierTag,'Enable',to);
       set(handles.unstableKuValueTag,'Enable','inactive');
       set(handles.calculatedUcritTag,'Enable','inactive');
       set(handles.calcuatedUTag,'Enable','inactive');
    end
else
   set(handles.unstableKuTag,'Enable',to);
   set(handles.criticalMultiplierTag,'Enable',to);
   set(handles.unstableKuValueTag,'Enable',to);
   set(handles.calculatedUcritTag,'Enable',to);
   set(handles.calcuatedUTag,'Enable',to);
end

set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);
set(handles.lenTag,'Enable',to);

set(handles.linearModelBtn,'Enable',to);
set(handles.nonlinearModelBtn,'Enable',to);


function nma_set_figure_position(the_handle,x,y,w,h)
%utility function, called to create a figure
%in middle of window
%
%by Nasser M. Abbasi
%

sz    = get(0,'ScreenSize');
wid   = sz(3);
hight = sz(4);
set(the_handle,'Units','pixels');
set(the_handle,'Position',[x*wid y*hight w*wid h*hight]);
```

Listing 4.2: lab4Main.m

```
function [g_msg,g_status]=lab4Main(data)
%called by GUI to implement the numerical solution for Lab4
%UC davis, spring 2011
%see lab4.m for the main GUI file which calls this function

%by Nasser M. Abbasi

g_msg    = '';
g_status =1;

%set up the initial conditions
IC = [data.vZero;        % v(0)
```

```
    data.rZero;         % r(0)
    data.thetaZero;     % theta(0)
    0;                  % X(0)
    0];                 % Y(0)

t = [0 data.maxt];       % simulation time

if get(data.handles.nonlinearModelBtn,'Value')==1
    [t,sol]  = ode15s(@rhsNONLINEAR,t,IC'); % numerically solve the system
else
    [t,sol]  = ode15s(@rhsLINEAR,t,IC'); % numerically solve the system
end
generatePlots(t,sol,data);     % plot the solution

% DONE that is all.

%------------
% Numerical solver, non-linear equations of motion
% This solver solves the set of equations of motion for the car
% dynamic model using the non-linear version of the equations.
% see report for the equations description
%------------
    function dy=rhsNONLINEAR(t,y)
        %order is  V,r,theta,X,Y

        theta    = y(3);
        r        = y(2);
        v        = y(1);
        steering = data.deltaZero*sin(2*pi*data.f*t);

        alphaf   = steering-atan ( (v+data.a*r) / data.U  );
        alphar   = atan ( (data.b*r-v) / data.U );
        Yf       = data.Cf*alphaf;
        Yr       = data.Cr*alphar;

        dy       = zeros(5,1);

        dy(1)    = (Yf/data.m)*cos(steering)+Yr/data.m-data.U * r;      %V
        dy(2)    = (data.a*Yf/data.Ic)*cos(steering)-data.b*Yr/data.Ic; %r
        dy(3)    = r;                                                   %theta
        dy(4)    = data.U*cos(theta)-v*sin(theta);                      %X
        dy(5)    = data.U*sin(theta)+v*cos(theta);                      %Y
    end

%------------
% Numerical solver, linear equations of motion
% This solver solves the set of equations of motion for the car
% dynamic model using the linear version of the equations.
% see report for the equations description
%------------
    function dy=rhsLINEAR(t,y)
        %order is  V,r,theta,X,Y

        alphaf=data.deltaZero*sin(2*pi*data.f*t)-...
            (y(1)+data.a*y(2))/data.U ;

        alphar=(data.b*y(2)-y(1))/data.U;

        Yf = data.Cf*alphaf;
        Yr= data.Cr*alphar;
```

```
        dy=zeros(5,1);

        dy(1)=Yf/data.m + Yr/data.m - data.U * y(2);  %V
        dy(2)=data.a*Yf/data.Ic - data.b*Yr/data.Ic;  %r
        dy(3)=y(2);                                   %theta
        dy(4)=data.U*cos(y(3))-y(1)*sin(y(3));        %X
        dy(5)=data.U*sin(y(3))+y(1)*cos(y(3));        %Y
    end
end

%--------------------------------
% This function is called after the solver is completed
% to generate the simulation and plots to the GUI
%--------------------------------
function generatePlots(t,sol,data)

%read the solutions from the sol matrix

v=sol(:,1);
r=sol(:,2);
maxR=max(r);
minR=min(r);
if abs(maxR-minR)<2*eps
    maxR=1;
    minR=-1;
end

theta=sol(:,3);
X=sol(:,4);
Y=sol(:,5);

MPH=(60*60)/5280;
SCREEN_CAPTURE=false;  %set to zero for animation capture

if SCREEN_CAPTURE
    frameNumber = 0;
    actualFrameNumber = 0;
    theFrame=getframe(gcf);
    [im,MAP]=rgb2ind(theFrame.cdata,32,'nodither');
end

b=data.b;
a=data.a;
L=data.L;

%------- plot global trajectory, reset plots
set(0,'CurrentFigure',data.handles.figure1);
cla(data.handles.mainAxes,'reset');
cla(data.handles.fixedBodyAxes,'reset');
cla(data.handles.tireForceAxes,'reset');
cla(data.handles.YrAxes,'reset');
cla(data.handles.slipAngleAxes,'reset');
cla(data.handles.slipAngleRAxes,'reset');
cla(data.handles.rVsTimeAxes,'reset');

maxX=max(X);
minX=min(X);
maxY=max(Y);
minY=min(Y);

maxX=maxX+0.05*abs(maxX);
```

```
minX=min(X)-0.05*abs(minX);
if minX==maxX
    minX=-1;
    maxX=1;
end

maxY=max(Y)+0.05*abs(maxY);
minY=min(Y)-0.05*abs(minY);
if minY==maxY
    minY=-1;
    maxY=1;
end

tireWidth=b/8;
tireLength=b/2;

leftTireX=[-b -b-tireLength -b-tireLength -b];
leftTireY=[tireWidth/2 tireWidth/2 -tireWidth/2 -tireWidth/2];

frontLeftTireX=[-tireLength/2 -tireLength/2 tireLength/2 tireLength/2];
frontLeftTireY=[-tireWidth/2 tireWidth/2 tireWidth/2 -tireWidth/2];

C=.5*ones(1,length(leftTireY));
carWidth=a/2;

bodyFrameX=[-.8*b -.8*b .5*a .65*a .5*a];
bodyFrameY=[-carWidth/2 carWidth/2 carWidth/2 0 -carWidth/2];

set(data.handles.figure1, 'CurrentAxes',data.handles.mainAxes);
xlim([minX maxX]);
ylim([minY maxY]);
grid;

%graphics layout
uLine=[0 0.25*a 0.1*a 0.25*a  0.1*a; ...
    0 0      0.1*a  0      -0.1*a];

vLinePositive=[0 0      -0.1*a    0      0.1*a; ...
    0 0.5*a   0.35*a   0.5*a  0.35*a];

vLineNegative=[0 0      -0.1*a    0        0.1*a; ...
    0 -0.5*a -0.35*a   -0.5*a  -0.35*a];

[Yf,Yr,maxYY,minYY,alphaF,alphaR]=getTireForces(t,v,r,data);
max_Yf=max(abs(Yf));
max_Yr=max(abs(Yr));
%normalized_Yf=Yf/max_Yf;
%normalized_Yr=Yr/max_Yr;

set(data.handles.figure1, 'CurrentAxes',data.handles.tireForceAxes);
xlim([0 t(end)]);
ylim([minYY maxYY]);
grid;

set(data.handles.figure1, 'CurrentAxes',data.handles.YrAxes);
xlim([0 t(end)]);
ylim([minYY maxYY]);
grid;

%update r vs time plot, do this here before start the animation
set(data.handles.figure1, 'CurrentAxes',data.handles.rVsTimeAxes);
```

```
plot(r(1:50:end)*180/pi,'r.-','LineWidth',1);
%plot(r*180/pi,'r.-','LineWidth',1);
title('r(t) vs time');
xlabel('time (sec)');
ylabel('yaw rate (deg/sec)');
hold on;
grid on;

%set(gca,'XTickLabel',[0 20 40 60 80 100]);
%xlim([0 t(end)]);
%ylim([minR-(abs(0.1*minR)) maxR+abs(0.1*maxR)]);
set(gca,'FontSize',7);


%set up various coordinates for the arrows to draw on the car as simulation
%is running.  Then use transformation to draw them
%negative is down, color red
negativeForceArrowOnBackWheel=...
    [-b-tireLength/2     -b-tireLength/2   -b-tireLength/2-a/10  ...
    -b-tireLength/2     -b-tireLength/2+a/10;

    -tireWidth/2        -tireWidth/2-0.3*a   -tireWidth/2-0.2*a  ...
    -tireWidth/2-0.3*a   -tireWidth/2-0.2*a];

positiveForceArrowOnBackWheel=...
    [-b-tireLength/2-a/10  -b-tireLength/2   -b-tireLength/2+a/10 ...
    -b-tireLength/2  -b-tireLength/2;

    -tireWidth/2-0.2*a    -tireWidth/2      -tireWidth/2-0.2*a  ...
    -tireWidth/2  -tireWidth/2-0.4*a];

%front force arrows
negativeForceArrowOnFrontWheel=...
    [0                    0                      -a/10       ...
    0                    a/10;

    -tireWidth/2        -tireWidth/2-0.3*a   -tireWidth/2-0.2*a ...
    -tireWidth/2-0.3*a   -tireWidth/2-0.2*a];

positiveForceArrowOnFrontWheel=...
    [-a/10                  0                     a/10      ...
    0                    0;

    -tireWidth/2-0.2*a    -tireWidth/2      -tireWidth/2-0.2*a  ...
    -tireWidth/2  -tireWidth/2-0.4*a];

%Now process the solution we allready have, and update plots and
%gui for each time step
for i=2:length(t)

    %plot global trajectory
    set(data.handles.figure1, 'CurrentAxes',data.handles.mainAxes);
    hold on;
    line(X(i-1:i),Y(i-1:i),'LineWidth',1.5,'LineStyle','-',...
        'color','red');
    title('global car motion view');
    %ylabel('Y'); xlabel('X');
    set(gca,'FontSize',7);

    set(data.handles.figure1, 'CurrentAxes',...
        data.handles.fixedBodyAxes);
```

```
cla;
line([-b*cos(theta(i))-carWidth/2*sin(theta(i)), ...
    a*cos(theta(i))-carWidth/2*sin(theta(i))],...
    [-b*sin(theta(i))+carWidth/2*cos(theta(i)),...
    a*sin(theta(i))+carWidth/2*cos(theta(i))],...
    'LineWidth',1.5,'LineStyle','-',...
    'color','red');

line([-b*cos(theta(i))+carWidth/2*sin(theta(i)), ...
    a*cos(theta(i))+carWidth/2*sin(theta(i))],...
    [-b*sin(theta(i))-carWidth/2*cos(theta(i)),...
    a*sin(theta(i))-carWidth/2*cos(theta(i))],...
    'LineWidth',1.5,'LineStyle','-',...
    'color','red');

line([-L L ],...
    [0 0 ],...
    'LineWidth',0.5,'LineStyle','-.',...
    'color','black');

line([0 0],...
    [-L L],...
    'LineWidth',0.5,'LineStyle','-.',...
    'color','black');

%Now do back wheel
%rotate back wheel at orgin
A = [cos(theta(i)) -sin(theta(i));sin(theta(i)) cos(theta(i))];
rotatedCoordinates=A*[ leftTireX;leftTireY];

%move from  0,0 to left back
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)+...
    carWidth/2*cos(theta(i));

rotatedCoordinates(1,:)=rotatedCoordinates(1,:)-...
    carWidth/2*sin(theta(i));

patch(rotatedCoordinates(1,:),rotatedCoordinates(2,:),C);

%move from  0,0 to right back
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)-carWidth*cos(theta(i));
rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+carWidth*sin(theta(i));
patch(rotatedCoordinates(1,:),rotatedCoordinates(2,:),C);

%add force arrow to left wheel
if Yr(i)<0 %CHECK
    forceArrow=negativeForceArrowOnBackWheel;
    lineColor='red';
else
    forceArrow=positiveForceArrowOnBackWheel;
    lineColor='blue';
end

%move for force arrow to right back tire only
rotatedCoordinates=A*[ forceArrow(1,:);forceArrow(2,:)];
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)-...
    carWidth/2*cos(theta(i));

rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+...
    carWidth/2*sin(theta(i));
```

```
line(rotatedCoordinates(1,:),rotatedCoordinates(2,:),...
    'color',lineColor);

%put text on the end of the arrow of the force amount
if Yr(i)<0 %CHECK
    text(rotatedCoordinates(1,2)+0.1*a*cos(theta(i)),...
        rotatedCoordinates(2,2)-0.1*a*cos(theta(i)),...
        sprintf('%3.3f',abs(Yr(i))),'interpreter','latex');
else
    text(rotatedCoordinates(1,5)+0.1*a*cos(theta(i)),...
        rotatedCoordinates(2,5)-0.1*a*cos(theta(i)),...
        sprintf('%3.3f',abs(Yr(i))),'interpreter','latex');
end

%do front    wheels
steeringAngle= data.deltaZero*sin(2*pi*t(i)*data.f);
B = [cos(theta(i)+ steeringAngle) -sin(theta(i)+ steeringAngle);...
    sin(theta(i)+ steeringAngle) cos(theta(i)+ steeringAngle)];

rotatedCoordinates=B*[  frontLeftTireX; frontLeftTireY];
%move the tire to the right front
rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+a*cos(theta(i));
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)+a*sin(theta(i));

%move the tire to the left front
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)+...
    carWidth/2*cos(theta(i));

rotatedCoordinates(1,:)=rotatedCoordinates(1,:)-...
    carWidth/2*sin(theta(i));

patch(rotatedCoordinates(1,:),rotatedCoordinates(2,:),C);

%right front
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)-carWidth*cos(theta(i));
rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+carWidth*sin(theta(i));
patch(rotatedCoordinates(1,:),rotatedCoordinates(2,:),C);

%add force arrow to right front wheel
if Yf(i)<0  %%CHECK
    forceArrow=negativeForceArrowOnFrontWheel;
    lineColor='red';
else
    forceArrow=positiveForceArrowOnFrontWheel;
    lineColor='blue';
end

%move for force arrow to right front tire only
rotatedCoordinates=B*[ forceArrow(1,:);forceArrow(2,:)];
rotatedCoordinates(2,:)=rotatedCoordinates(2,:)+a*sin(theta(i));
rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+a*cos(theta(i));

rotatedCoordinates(2,:)=rotatedCoordinates(2,:)-carWidth/2*cos(theta(i));
rotatedCoordinates(1,:)=rotatedCoordinates(1,:)+carWidth/2*sin(theta(i));

line(rotatedCoordinates(1,:),rotatedCoordinates(2,:),'color',lineColor);

%put text on the end of the arrow of force amount
if Yf(i)<0 %CHECK
    text(rotatedCoordinates(1,2)+0.05*a*cos(theta(i)),...
        rotatedCoordinates(2,2)-0.05*a*cos(theta(i)),...
```

```
        sprintf('%3.3f',abs(Yf(i))),'interpreter','latex');
else
    text(rotatedCoordinates(1,5)+0.05*a*cos(theta(i)),...
        rotatedCoordinates(2,5)-0.05*a*cos(theta(i)),...
        sprintf('%3.3f',abs(Yf(i))),'interpreter','latex');
end

%do centered frame U , V arrow lines
rotatedCoordinates=A*[  bodyFrameX; bodyFrameY];
patch(rotatedCoordinates(1,:),rotatedCoordinates(2,:),...
    0.5*ones(1,length( bodyFrameX)));

rotatedCoordinates=A*uLine;
line(rotatedCoordinates(1,:),rotatedCoordinates(2,:),'color','black');
%add value of U on top of arrow
text(rotatedCoordinates(1,2)+0.1*a*cos(theta(i)),...
    rotatedCoordinates(2,2)+0.1*a*sin(theta(i)),...
    sprintf('%3.1f',data.U*MPH),'interpreter','latex');

%check if velocity (lateral) on the car is positive or negative
if v(i)<=0
    rotatedCoordinates=A*vLineNegative;
    lineColor='red';
    delX=0.1*a*sin(theta(i));
    delY=-0.1*a*cos(theta(i));
else
    rotatedCoordinates=A*vLinePositive;
    lineColor='black';
    delX=-0.1*a*sin(theta(i));
    delY=0.1*a*cos(theta(i));
end

line(rotatedCoordinates(1,:),rotatedCoordinates(2,:),'color',lineColor);
%add value of V on top of arrow
text(rotatedCoordinates(1,2)+delX,rotatedCoordinates(2,2)+delY,...
    sprintf('%3.2f',abs(v(i))*MPH),'interpreter','latex');

title('body fixed coordinates view');
if get(data.handles.overSteerBtn,'Value')==1
    if abs(alphaF(i))*180/pi > 8
        if abs(alphaF(i))*180/pi > 15
            text(-6,6,...
                'CRITICAL! front slip angle out of control, UNSTABLE!',...
                'FontSize',10,'Color','red')
        else
            text(-6,6,...
                'WARNING!  front tire slip angle too large','FontSize',...
                10,'Color','red')
        end
    else
        if abs(alphaR(i))*180/pi > 8
            if abs(alphaF(i))*180/pi > 15
                text(-6,6,...
                    'CRITICAL!  rear slip angle out of control, UNSTABLE!',...
                    'FontSize',10,'Color','red')
            else
                text(-6,6,...
                    'WARNING! back tire slip angle too large',...
                    'FontSize',10,'Color','red')
            end
        end
```

```
    end
end

%add resultant speed
alpha=atan(v(i)/data.U);
valueOfSpeed=sqrt(v(i)^2+data.U^2);

%lengthOfSpeed=1.5*b; %sqrt((0.25*a)^2+(0.5*a)^2);
%Xcoord=lengthOfSpeed*cos(alpha);
%Ycoord=lengthOfSpeed*sin(alpha);

CoordinatesOfArrow=[0 1.5*a  1.4*a 1.5*a 1.4*a;...
    0 0 0.1*a 0 -0.1*a];
comAngle=alpha+theta(i);
B = [cos(comAngle) -sin(comAngle);...
    sin(comAngle) cos(comAngle)];

rotatedCoordinates=B*[CoordinatesOfArrow(1,:);CoordinatesOfArrow(2,:)];
line(rotatedCoordinates(1,:),rotatedCoordinates(2,:),...
    'color','magenta',...
    'LineWidth',1.5,'LineStyle','-');

if rotatedCoordinates(2,2)<=0
    delX=0.1*a;
    delY=-0.1*a;
else
    delX=-0.1*a;
    delY=0.1*a;
end

text(rotatedCoordinates(1,2)+delX,rotatedCoordinates(2,2)+delY,...
    sprintf('%3.2f mph',valueOfSpeed*MPH),'interpreter','latex');

xlim([-data.L data.L]);
ylim([-data.L data.L]);
axis equal;
axis tight;
hold off;
drawnow;

%tire forces plot
set(data.handles.figure1, 'CurrentAxes',data.handles.tireForceAxes);
hold on;
plot(t(i-1:i),Yf(i-1:i),'k-','LineWidth',1);
title('Force on front tire vs. time(sec)');
%ylabel('Newton');
set(gca,'FontSize',7);
drawnow;

set(data.handles.figure1, 'CurrentAxes',data.handles.YrAxes);
hold on;
plot(t(i-1:i),Yr(i-1:i),'k-','LineWidth',1);
title('Force on rear tire vs. time(sec)');
%ylabel('Newton');
set(gca,'FontSize',7);

%make slip angle plot
set(data.handles.figure1, 'CurrentAxes',data.handles.slipAngleAxes);
hold on;
plot(alphaF(i-1:i)*180/pi,Yf(i-1:i),'r-','LineWidth',1);
title('front tire force vs slip angle');
```

```
    %xlabel('slip angle in deg');
    %ylabel('Yf (lb)');

    ylim([-10*data.Cf*pi/180 10*data.Cf*pi/180]);
    xlim([-10 10]);
    set(gca,'FontSize',7);

    %updateSimulationOutput_lab4_eme121(data.handles,t(i),v(i),...
    %    r(i),Yf(i),Yr(i),theta(i),valueOfSpeed,data.Cf,data.Cr,...
    %    alphaF(i),alphaR(i),steeringAngle)  ;
    %drawnow;

    set(data.handles.figure1, 'CurrentAxes',data.handles.slipAngleRAxes);
    hold on;
    plot(alphaR(i-1:i)*180/pi,Yr(i-1:i),'r-','LineWidth',1);
    title('rear tire force vs slip angle');
    %xlabel('slip angle in deg');
    %ylabel('Yr (lb)');

    ylim([-10*data.Cr*pi/180 10*data.Cr*pi/180]);
    xlim([-10 10]);
    set(gca,'FontSize',7);


    updateSimulationOutput_lab4_eme121(data.handles,t(i),v(i),...
        r(i),Yf(i),Yr(i),theta(i),valueOfSpeed,data.Cf,data.Cr,...
        alphaF(i),alphaR(i),...
        steeringAngle)  ;
    drawnow;

    if SCREEN_CAPTURE
        frameNumber = frameNumber+1;
        if mod(frameNumber,1)==0
            theFrame = getframe(gcf);
            actualFrameNumber = actualFrameNumber +1;
            im(:,:,1,actualFrameNumber ) = ...
                rgb2ind(theFrame.cdata,MAP,'nodither');
            %im=rgb2ind(theFrame.cdata,MAP,'nodither');
            %imwrite(im,MAP,[num2str(actualFrameNumber) '.gif'],'gif');
        end
    end

    userData = get(data.handles.figure1, 'UserData');
    if userData.stop
        break;
    end
end

if SCREEN_CAPTURE
    imwrite(im,MAP,'demo.gif','DelayTime',0,'LoopCount',inf,...
        'WriteMode','overwrite');
end

end
%--------------------------
% Called by the plotting function above to calculate the tire
% forces Yf, Yr from the result of the numerical solution
% so we can plot them.
%--------------------------
function [Yf,Yr,maxY,minY,alphaF,alphaR]=getTireForces(t,v,r,data)
```

```
Yf=zeros(1,length(t));
Yr=Yf;
alphaR=Yr;
alphaF=Yr;

for i=1:length(t)
    alphaF(i)=data.deltaZero*sin(2*pi*t(i)*data.f)-...
        (v(i)+data.a*r(i))/data.U;

    Yf(i)=data.Cf*alphaF(i);
    alphaR(i)=(data.b*r(i)-v(i))/data.U;
    Yr(i)=data.Cr*alphaR(i);
end
maxY=max([max(Yf),max(Yr)]);
minY=min([min(Yf),min(Yr)]);
if abs(minY-maxY)<2*eps
    minY=-1;
    maxY=1;
end

end

%--------------------------
% Called by the plotting function above to update the GUI
% at each time step with current simulation variables
%--------------------------
function updateSimulationOutput_lab4_eme121(...
    handles,t,v,r,Yf,Yr,theta,valueOfSpeed,Cf,Cr,alphaF,alphaR,...
    steeringAngle)

MPH=(60*60)/5280;
FTS=1/MPH;

set(handles.timeValueTag,'String',sprintf('%3.3f',t));
set(handles.speedValueTag,'String',sprintf('%3.3f',valueOfSpeed*MPH));
set(handles.yawRateValueTag,'String',sprintf('%3.3f',r*180/pi));

set(handles.YfValueTag,'String',sprintf('%3.3f',Yf));
set(handles.YrValueTag,'String',sprintf('%3.3f',Yr));
set(handles.thetaValueTag,'String',sprintf('%3.3f',mod(theta*180/pi,360)));

set(handles.CfValueTag,'String',sprintf('%3.3f',Cf*pi/180));
set(handles.CrValueTag,'String',sprintf('%3.3f',Cr*pi/180));
set(handles.steerAngleValueTag,'String',sprintf('%3.3f',steeringAngle*180/pi));


set(handles.alpha_f_valueTag,'String',sprintf('%3.3f',...
    sign(alphaF)*mod(abs(alphaF)*180/pi,360)));

set(handles.alpha_r_valueTag,'String',sprintf('%3.3f',...
    sign(alphaR)*mod(abs(alphaR)*180/pi,360)));

set(handles.V_valueTag,'String',sprintf('%3.3f',v*MPH));

end
```

4.5 Lab 5,simulation of motion of a 2 degree of freedom car trailer

4.5.1 Problem description

Simulation 5 Two Degree-of-Freedom Trailer

In class we discussed the stability of a trailer shown schematically in the figure below.

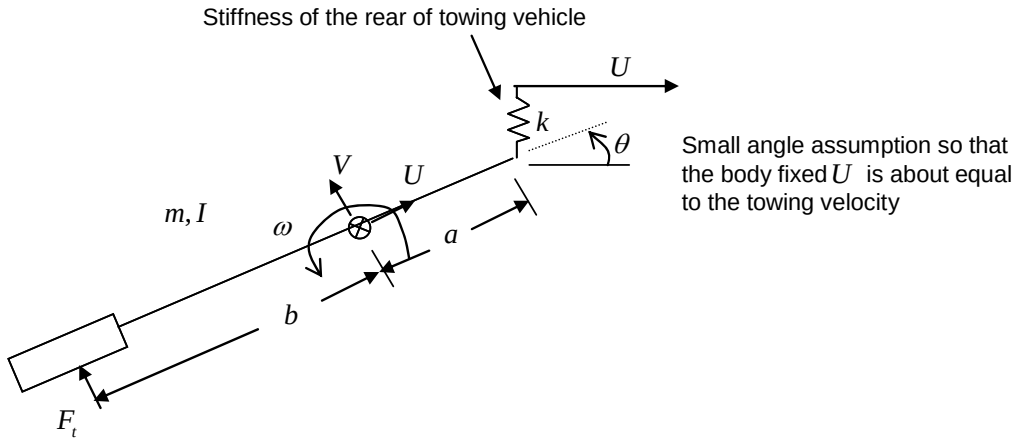


Figure 1 Schematic of the 2 d-o-f trailer

This simulation is to see if the analysis done in class is supported by simulation of the equations of motion. In class we determined that for the "no side slip" condition at the rear of the trailer, stability required,
$I < mab$.

The parameters for the trailer are:

```
U=50*.45;          Forward velocity, m/s
m=1800/2.2;        trailer mass, Kg
L=7*.3048;         trailer length, m
ab=.85;            a/b ratio
b=L/(1+ab);        Calculation of b
a=L-b;             Calculation of a
I=gain*m*a*b;      .8<I<1.2, a range that brackets the stability
condition
C_t=210.*4.448*180/pi; Cornering stiffness at rear of trailer, N/rad
k=m*(2*pi*2)^2;    stiffness of front spring. You might try different
values
```

Your job:
A. Derive the equations of motion any way you desire and show that you get these first order equations.

$$\frac{dV}{dt} = \frac{k}{m}\delta_s + \frac{F_t}{m} - Ur$$
$$\frac{dr}{dt} = -\frac{F_t b}{I} + \frac{ka}{I}\delta_s$$
$$\frac{d\theta}{dt} = r$$
$$\frac{d\delta_s}{dt} = -(U\theta + V + ar)$$

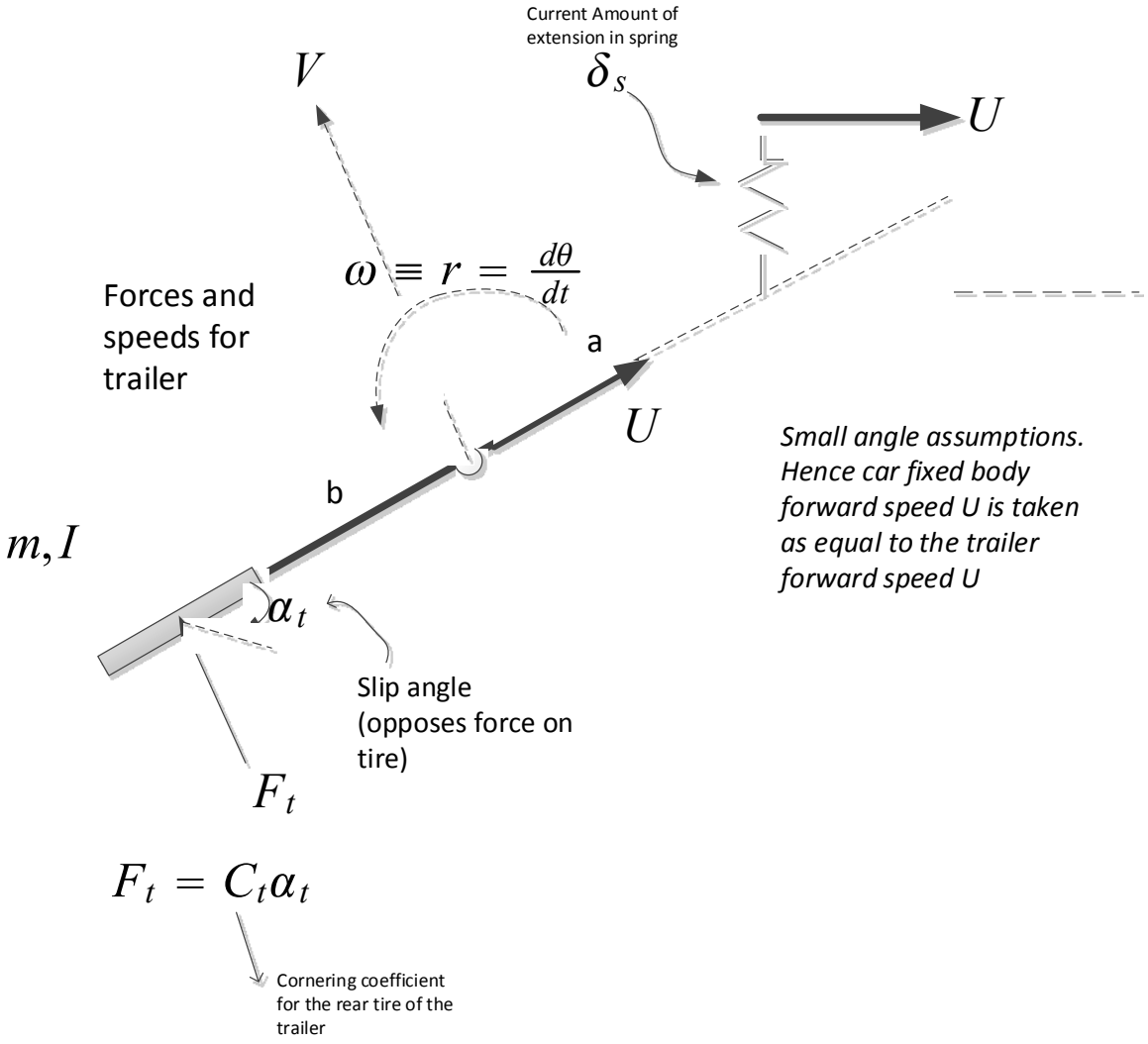
where

$$\alpha_t = \frac{br - V}{U}$$
$$F_t = C_t \alpha_t$$

B. Set up a simulation of this system and plot results that demonstrate whether or not the analysis in class is supported by the simulation where the no slip condition is not used.

4.5.2 Part A. Derivation of equations of motion (slip condition)

The following diagram shows the forces on trailer

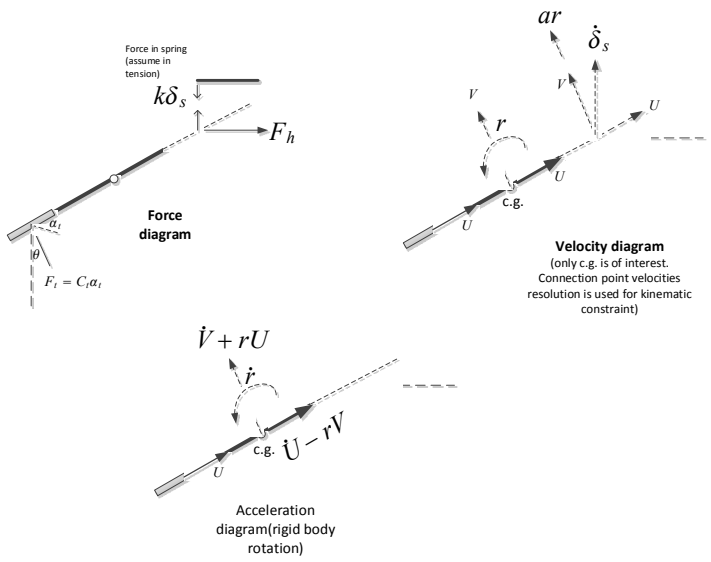


To derive the equations of motion for the trailer body, we use $F = ma$ and then resolve forces in the vertical direction and horizontal directions along the fixed body coordinates centered on the c.g. of the trailer. We also use the moment equation on the trailer as it is a rigid body.

In the following analysis, the following assumptions are made

1. Angle θ is small
2. Force F_h is small
3. Spring displacement δ_s is small
4. U is constant
5. slip angle α_t is small
6. the no-slip condition is *not used* in the derivation. In other words, we can no longer assume that $V = br$ as we did in the class problem.

To be able to use $F = ma$, we need to first make the velocity, acceleration and force diagrams. We are only interested in forces, velocity and acceleration of the c.g. of the trailer, since that is the body we will generate equations of motions for. The car itself will not be involved. We will also use kinematics constraint at connection point between the trailer and the car in order to generate an extra equation to solve for the unknown spring displacement δ_s as will be shown below.



Resolving forces In the vertical direction (vertical relative to fixed body coordinates) we obtain

$$\sum F_y = ma_y$$
$$F_t + k\delta_s \cos\theta - F_h \sin\theta = m\left(\dot{V} + rU\right)$$

Resolving forces In the horizontal direction gives

$$\sum F_x = ma_x$$
$$k\delta_s \sin\theta + F_h \cos\theta = m\left(\dot{U} - rV\right)$$

Taking moments around an axis through the c.g. of the trailer gives

$$\sum M = I_g\dot{r}$$
$$(k\delta_s \cos\theta)\, a - F_t b - F_h \sin\theta a = I_g\dot{r}$$

It is also clear that (by definition) that the yaw rate r is related to the angle θ by

$$r = \frac{d\theta}{dt}$$

To solve for δ_s, which is an unknown in the above equations, we need an additional equation. We are run out of equations using $F = ma$, so we must resolute to using a kinematic (constraint) equation. At

the point of the connection between the trailer and the car, the speed of the spring displacement as seen by the car, must be the same as the speed of the connection as seen by the trailer. Hence this gives

$$\frac{d\delta_s}{dt} = -\left(V\cos\theta + U\sin\theta + ar\right)$$

Therefore, we have a total of 5 equations

$$\begin{aligned}
F_t + k\delta_s\cos\theta - F_h\sin\theta &= m\left(\dot{V} + rU\right) && (1)\\
k\delta_s\sin\theta + F_h\cos\theta &= m\left(\dot{U} - rV\right) && (2)\\
\left(k\delta_s\cos\theta\right)a - F_t b - F_h\sin\theta a &= I_g\dot{r} && (3)\\
r &= \dot{\theta} && (4)\\
\dot{\delta}_s &= -\left(V\cos\theta + U\sin\theta + ar\right) && (5)
\end{aligned}$$

Now we apply the assumptions that θ is small, hence the above reduces to

$$\begin{aligned}
F_t + k\delta_s - F_h\theta &= m\left(\dot{V} + rU\right) && (1)\\
k\delta_s\theta + F_h &= m\left(\dot{U} - rV\right) && (2)\\
k\delta_s a - F_t b - F_h\theta a &= I_g\dot{r} && (3)\\
r &= \dot{\theta} && (4)\\
\dot{\delta}_s &= -\left(V + U\theta + ar\right) && (5)
\end{aligned}$$

Another assumption is thatF_h is very small, and also δ_s is very small. Hence the products $F_h\theta$ and $\delta_s\theta$ vanish from the above equations resulting in

$$\begin{aligned}
F_t + k\delta_s &= m\left(\dot{V} + rU\right) && (1)\\
F_h &= m\left(\dot{U} - rV\right) && (2)\\
k\delta_s a - F_t b &= I_g\dot{r} && (3)\\
r &= \dot{\theta} && (4)\\
\dot{\delta}_s &= -\left(V + U\theta + ar\right) && (5)
\end{aligned}$$

Now we apply the assumption that U is constant, hence $\dot{U} = 0$ and hence equation (2) above can be removed as it provides no information. In the above, $F_t = C_t\alpha_t$, and $\alpha_t = \tan^{-1}\left(\frac{br-V}{U}\right)$. Hence $F_t = C_t\alpha_t = C_t\tan^{-1}\left(\frac{br-V}{U}\right) = C_t\frac{br-V}{U}$ assuming small slip angle.

Therefore, the final equations are

$$F_t = C_t \frac{br - V}{U} \tag{1A}$$

$$\dot{V} = \frac{F_t}{m} + \frac{k\delta_s}{m} - rU \tag{2A}$$

$$\dot{r} = -\frac{F_t b}{I_g} + \frac{k\delta_s a}{I_g} \tag{3A}$$

$$\dot{\theta} = r \tag{4A}$$

$$\dot{\delta}_s = -(V + U\theta + ar) \tag{5A}$$

And the above equations is what will be solved to generate V, r, θ, δ_s. This completes the first part of the assignment.

Part A. Extra. Derivation of equations of motion (no-slip condition)

In addition to the above, the no-slip condition case was implemented in order to be able to compare the result from the above against. The no-slip condition implies $V = br$. Hence $F_t = 0$. Now, solving for F_t from equation (2A) above, and substitute the result into (3A) this gives

$$m\left(\dot{V} + rU\right) - k\delta_s = F_t$$

Hence (3A) becomes

$$\dot{r} = -\frac{\left(m\left(\dot{V} + rU\right) - k\delta_s\right) b}{I_g} + \frac{k\delta_s a}{I_g}$$
$$\dot{r} I_g = -mb\left(\dot{V} + rU\right) - bk\delta_s + k\delta_s a$$

But $\dot{V} = b\dot{r}$ hence the above becomes

$$\dot{r}\left(I_g + mb^2\right) + r\left(mbU\right) - (b + a)\, k\delta_s = 0$$

Therefore, for the no-slip condition, the following are the 3 equations to solve

$$\dot{r} = \frac{-r\left(mbU\right) + (b + a)\, k\delta_s}{I_g + mb^2} \tag{1B}$$

$$\dot{\theta} = r \tag{2B}$$

$$\dot{\delta}_s = -(V + U\theta + ar) \tag{3B}$$

Now the slip and the no-slip conditions can be compared using the user interface more easily.

4.5.3 Part 2

In the class, we found that, when the no-slip assumption was used (i.e. $V = br$), then the trailer was stable as long as $I_g < mab$. In this assignment, we are asked not use the no-slip condition, and the goal now is to find if the trailer will become unstable as well when $I_g < mab$ and if it will become stable when $I_g \geq mab$.

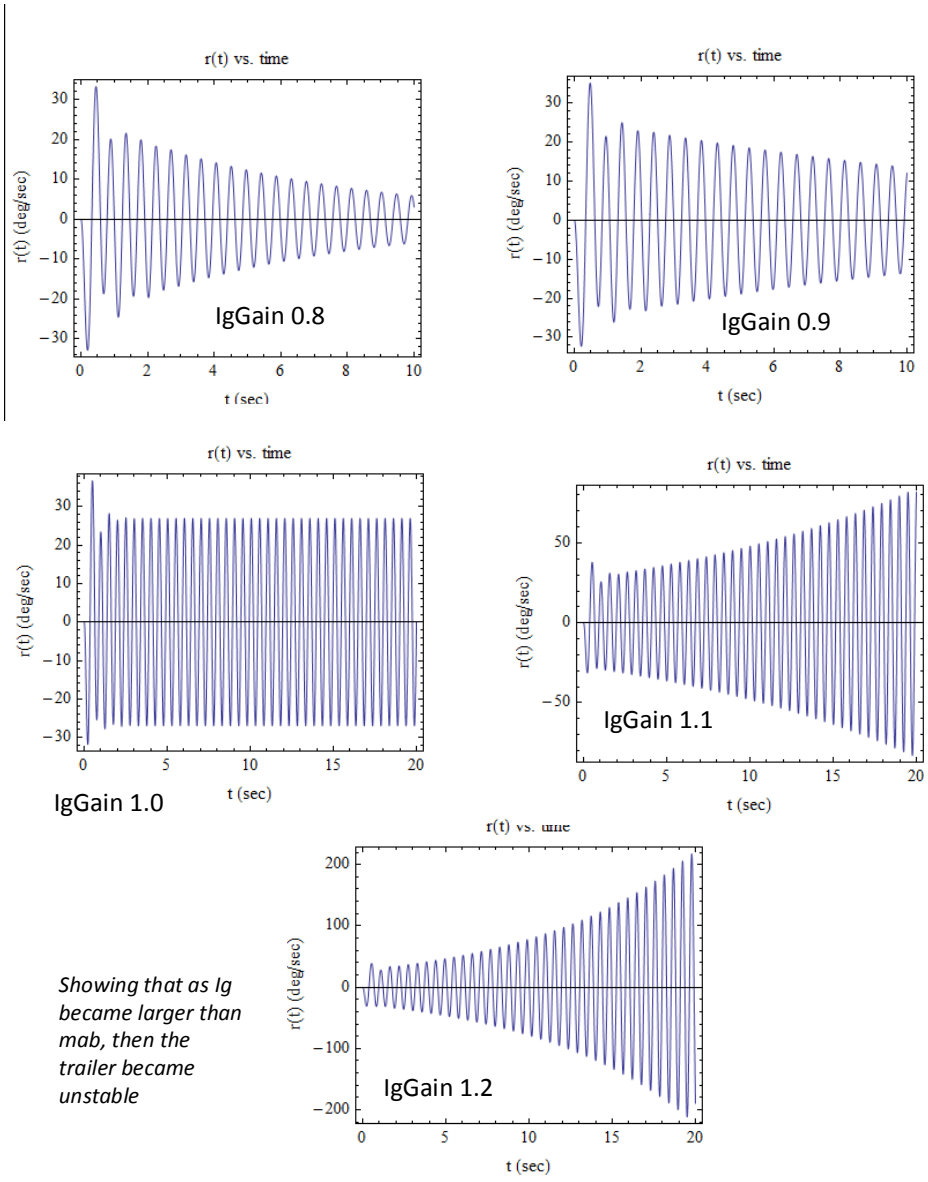
The simulation was run for the following values of I_g

$$I_g = [0.8, 0.9, 1.0, 1.1, 1.2]\, mab$$

The results are shown below, followed by a conclusion. The simulation was run for around 20 seconds in order to obtain a plot that looked reasonable.

Results of simulation (slip condition on)

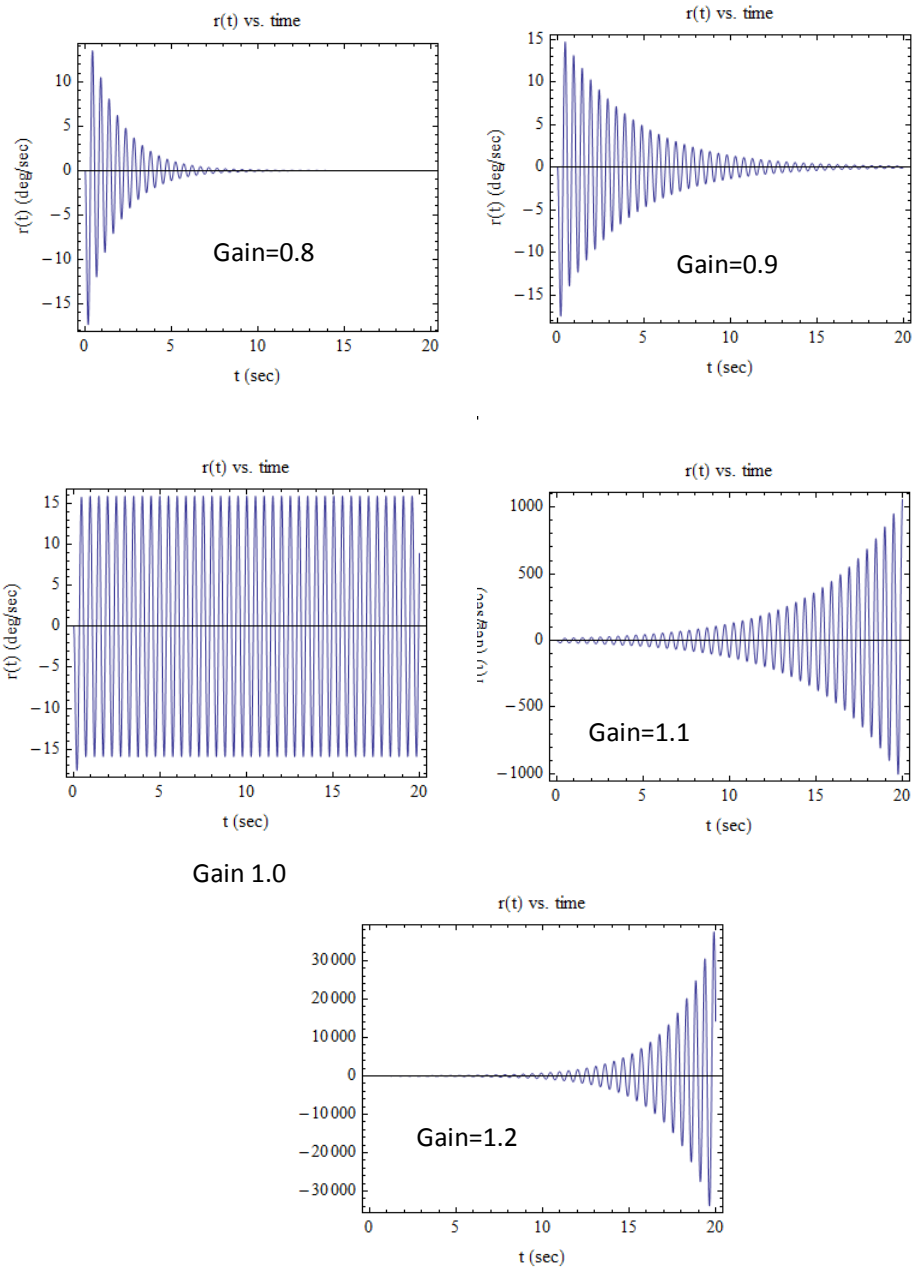
In all of these results, the yaw rate ($r(t)$) is shown as the $I_g gain$ value is increased from 0.8 to 1.2. All other parameters are as specified in the handout.



Showing that as Ig became larger than mab, then the trailer became unstable

Result of simulation (Extra, no-slip condition)

In addition to the slip-on model we did above, the no-slip condition was also simulated. We know that this condition will make the trailer unstable if $I_g > mab$, and this was done just for verification. The results are similar to the above.



4.5.4 Conclusion

The results of the simulation when the slip condition is applied shows that the trailer will become unstable when $I_g > mab$. This result matches the results obtained in class using stability analysis when the no-slip condition was used.

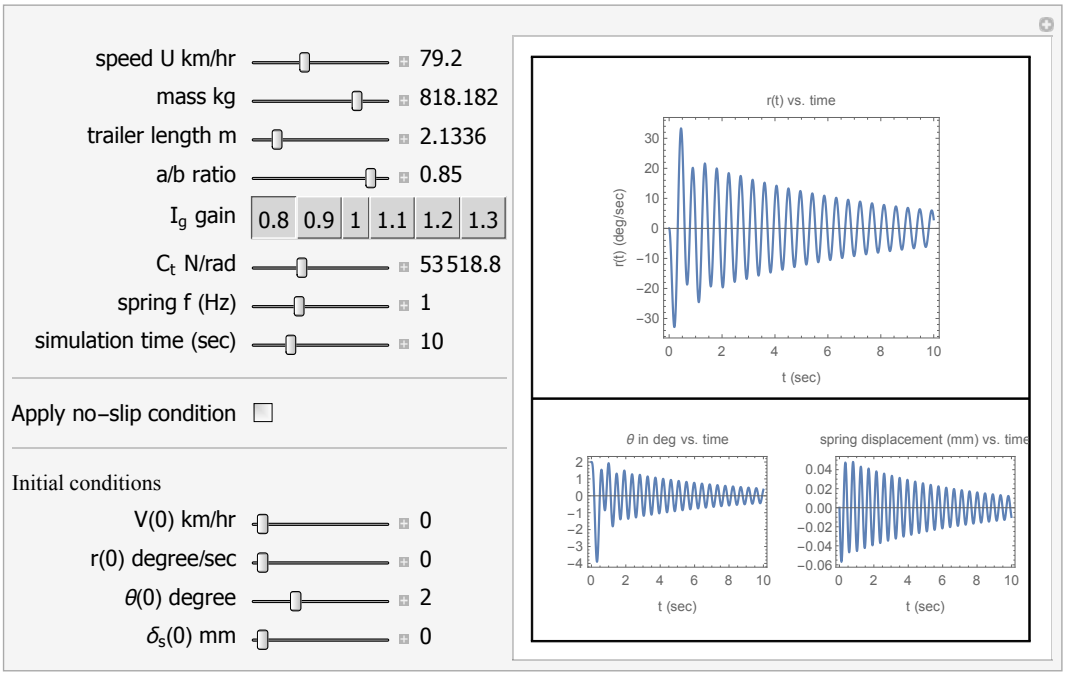
The only difference in results between using the no-slip (class analysis) and using the slip condition, is that, when $I_g > mab$, it took longer time for the trailer with the no-slip condition turned off, to reach the same yaw rate as when the no-slip condition is turned on (class). I.e. the trailer will in both cases become unstable, but the trailer become unstable more rapidly when the no-slip condition is used. The reason is that the tire force is no longer available to reduce the yaw movement, since when no-slip

condition is used, the force on the trailer tire is removed. Having this force there, the trailer will take longer time to reach the same yaw rate.

So, in conclusion, the trailer will become unstable when $I_g > mab$ and this simulation has verified the result found in class.

4.5.5 Appendix

Mathematica Used for the following simulation of a simple car trailer stability analysis.




```
Manipulate[
 process[u, m, L, abRatio, IgGain, Ct, v0, r0, \[Theta]0, \[Delta], f, maxt,
  noslip],
 {{u, 79.2, "speed U km/hr"}, 10, 200, 1, Appearance -> "Labeled",
  ImageSize -> Small},
 {{m, 1800/2.2, "mass kg"}, 1, 1000, 1, Appearance -> "Labeled",
  ImageSize -> Small},
 {{L, 7*0.3048, "trailer length m"}, 1, 10, 0.1, Appearance -> "Labeled",
  ImageSize -> Small},
 {{abRatio, 0.85, "a/b ratio"}, 0.1, 0.9, 0.1, Appearance -> "Labeled",
  ImageSize -> Small},
 {{IgGain, 0.8, "\!\(\*SubscriptBox[\(I\), \(g\)]\) gain"}, {0.8 -> 0.8,
   0.9 -> 0.9, 1 -> 1, 1.1 -> 1.1, 1.2 -> 1.2, 1.3 -> 1.3},
  ControlType -> SetterBar, ImageSize -> Small},
 {{Ct, 210*4.448*180/Pi, "\!\(\*SubscriptBox[\(C\), \(t\)]\) N/rad"}, 30000,
  100000, 1000, Appearance -> "Labeled", ImageSize -> Small},
 {{f, 1, "spring f (Hz)"}, 0.1, 3, 0.1, Appearance -> "Labeled",
  ImageSize -> Small},
 {{maxt, 10, "simulation time (sec)"}, 0.1, 40, 0.1, Appearance -> "Labeled",
  ImageSize -> Small},

 Delimiter,
```

```
Control[{{noslip, False, "Apply no-slip condition"}, {False, True},
  ControlType -> Checkbox, ImageSize -> Small}],

Delimiter,

Text["Initial conditions"],
{{v0, 0, "V(0) km/hr"}, 0, 10, 1, Appearance -> "Labeled",
 ImageSize -> Small},
{{r0, 0, "r(0) degree/sec"}, 0, 7, 1, Appearance -> "Labeled",
 ImageSize -> Small},
{{\[Theta]0, 2, "\[Theta](0) degree"}, 0, 7, .1, Appearance -> "Labeled",
 ImageSize -> Small},
{{\[Delta], 0, "\!\(\*SubscriptBox[\(\[Delta]\), \(s\)]\)(0) mm"}, 0, 100, 1,
  Appearance -> "Labeled", ImageSize -> Small},

SynchronousUpdating -> False,(*important to have this*)
ContinuousAction -> False,

Initialization :>
 (
  process[uu0_, m_, L_, abRatio_, IgGain_, c_, vv0_,
    rr0_, \[Theta]\[Theta]0_, \[Delta]\[Delta]0_, f_, maxt_, noslip_] :=
   Module[{b = 1./(1 + abRatio), a, k = m*(2.*Pi*f)^2, u0 = uu0*1000/3600.,
     v0 =
      vv0*1000/3600., \[Theta]0 = \[Theta]\[Theta]0*
       Pi/180., \[Delta]0 = \[Delta]\[Delta]0/1000., r0 = rr0*Pi/180., F,
     eq1, eq2, eq3, eq4, t, r, v, \[Theta], \[Delta], Ig, sol, data},

    a = L - b;
    F = c (b r[t] - v[t])/u0;
    Ig = IgGain m a b;

    If[noslip == False, {
      eq1 = v'[t] == F/m + (k \[Delta][t])/m - r[t] u0;
      eq2 = r'[t] == (k \[Delta][t] a )/Ig - (F b)/Ig;
      eq3 = \[Theta]'[t] == r[t];
      eq4 = \[Delta]'[t] == -(v[t] + u0 \[Theta][t] + a r[t]);
      sol =
       First@DSolve[{eq1, eq2, eq3, eq4, v[0] == v0,
          r[0] == r0, \[Theta][0] == \[Theta]0, \[Delta][
            0] == \[Delta]0}, {v[t], r[t], \[Theta][t], \[Delta][t]}, t]
      }, {
      eq1 = r'[t] == (-r[t] (m b u0) + (b + a) k \[Delta][t])/(Ig + m b^2);
      eq2 = \[Theta]'[t] == r[t];
      eq3 = \[Delta]'[t] == -(b r[t] + u0 \[Theta][t] + a r[t]);
      sol =
       First@
        DSolve[{eq1, eq2, eq3,
          r[0] == r0, \[Theta][0] == \[Theta]0, \[Delta][
            0] == \[Delta]0}, {r[t], \[Theta][t], \[Delta][t]}, t]
      }
     ];


    Grid[
     {
      {Plot[Chop@Evaluate[r[t] /. sol]*180/Pi, {t, 0, maxt},
        PlotRange -> All, Frame -> True,
        FrameLabel -> {{"r(t) (deg/sec)", None}, {"t (sec)",
           "r(t) vs. time"}}, ImagePadding -> {{40, 10}, {40, 40}},
        ImageSize -> 250, AspectRatio -> 0.8]
```

```
       },
      {
       Row[{
         Plot[Chop@Evaluate[\[Theta][t] /. sol]*180/Pi, {t, 0, maxt},
          PlotRange -> All, Frame -> True,
          FrameLabel -> {{None, None}, {"t (sec)",
             Row[{"\[Theta] in deg vs. time"}]}},
          ImagePadding -> {{40, 10}, {50, 40}}, ImageSize -> 180]
         ,
         Plot[Chop@Evaluate[\[Delta][t] /. sol], {t, 0, maxt},
          PlotRange -> All, Frame -> True,
          FrameLabel -> {{None, None}, {"t (sec)",
             Row[{"spring displacement (mm) vs. time"}]}},
          ImagePadding -> {{40, 10}, {50, 40}}, ImageSize -> 180]
         }]
       }
      }, Frame -> All, Spacings -> 0
     ]
    ]

  )
]
```